IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

SEVEN NETWORKS, LLC	
Plaintiff,	**Civil Action No. 2:19-cv-115**
v.	
APPLE INC.	**JURY TRIAL REQUESTED**
Defendant.	

COMPLAINT

Plaintiff SEVEN Networks, LLC ("SEVEN") files this complaint for patent infringement against Defendant Apple Inc. ("Apple") for infringement of U.S. Patent Nos. 9,369,539 ("the '539 patent"), 9,438,550 ("the '550 patent"), 9,473,914 ("the '914 patent"), 9,516,127 ("the '127 patent"), 9,603,056 ("the '056 patent"), 9,608,968 ("the '968 patent"), 9,648,557 ("the '557 patent"), 9,712,476 ("the '476 patent"), 9,712,986 ("the '986 patent"), 9,769,176 ("the '176 patent"), 10,027,619 ("the '619 patent"), 10,039,029 ("the '029 patent"), 10,091,734 ("the '734 patent"), 10,110,534 ("the '534 patent"), 10,135,771 ("the '771 patent"), 10,243,962 ("the '962 patent") (collectively the "patents-in-suit"), pursuant to 35 U.S.C. § 271.

I. PARTIES

1. Plaintiff SEVEN is a company formed under the laws of Delaware with its principal place of business at 2660 East End Boulevard South, Marshall, Texas 75672.

2. Defendant Apple is a California corporation with its principal place of business at 1 Infinite Loop, Cupertino, California 95014. Apple designs, manufactures, makes, uses,

imports into the United States, sells, and/or offers for sale in the United States Apple smartphones, tablets, smartwatches, macs, and Apple servers. Apple's smartphones, tablets, smartwatches, and macs are marketed, used, offered for sale, and/or sold throughout the United States, including within this district.

II. JURISDICTION AND VENUE

3. Within the United States, this Court has subject matter jurisdiction over this case under 28 U.S.C. §§ 1331, 1332, 1338, and 1367.

4. The amount in controversy exceeds $75,000.

5. Venue is proper in this Court pursuant to 28 U.S.C. §§ 1391 and 1400(b).

6. This Court has personal jurisdiction over Apple. Apple has continuous and systematic business contacts with the state of Texas. Apple, directly or through subsidiaries or intermediaries (including distributors, retailers, and others), conducts its business extensively throughout Texas, by shipping, distributing, making, using, offering for sale, selling, and advertising (including the provision of interactive web pages) its products and services in the state of Texas and the Eastern District of Texas. Apple, directly and through subsidiaries or intermediaries (including distributors, retailers, and others), has purposefully and voluntarily placed infringing products and services into this district and into the stream of commerce with the intention and expectation that they will be purchased and used by consumers in this district. Apple has offered and sold and continues to offer and sell these infringing products and services in this district, including at physical Apple stores located within this district. Apple and its customers also commit additional acts of direct infringement in this district with respect to each asserted patent through their infringing use of the accused devices, including Apple's servers, in this district, including when Apple and its customers put the accused devices into service and receive a benefit, and Apple is liable for these additional acts of direct infringement and indirect

infringement in this district. Apple has committed acts of infringement, both direct and indirect, in this district with respect to each asserted patent and has a regular and established place of business in this judicial district.

III. BACKGROUND

7. For nearly two decades, SEVEN has researched and developed innovative software solutions for mobile devices. For example, SEVEN has developed software technologies to manage mobile traffic in order to conserve network and battery resources.

8. SEVEN has been recognized in the industry for its innovative technologies and products. For example, at the Mobile World Congress in 2011, the GSMA awarded SEVEN with its Global Mobile Award for Best Technology Breakthrough. Further, in 2013 SEVEN won the Mobile Merit Award for its outstanding innovations in the mobile industry and was identified as one of fifty mobile companies to watch by AlwaysOn. SEVEN was also awarded the Best Free Android App in 2013 by TechRadar and Telecoms.com identified SEVEN in its Best LTE Traffic Management Product Short List.

IV. THE ASSERTED PATENTS

9. On June 14, 2016, the United States Patent and Trademark Office issued U.S. Patent No. 9,369,539 ("the '539 patent"), entitled "Method and Device for Power Saving for Downloading Files." Plaintiff is the assignee of all rights, title, and interest in and to the '539 patent and possesses all rights of recovery under the '539 patent, including the right to recover damages for present, past, and future infringement. The '539 patent is valid and enforceable.

10. On September 6, 2016, the United States Patent and Trademark Office issued U.S. Patent No. 9,438,550 ("the '550 patent"), entitled "Mobile Device Power Management in Data Synchronization Over a Mobile Network With or Without a Trigger Notification." Plaintiff is the assignee of all rights, title, and interest in and to the '550 patent and possesses all rights of

recovery under the ’550 patent, including the right to recover damages for present, past, and future infringement. The ’550 patent is valid and enforceable.

11. On October 18, 2016, the United States Patent and Trademark Office issued U.S. Patent No. 9,473,914 (“the ’914 patent”), entitled “System and Method for Providing a Network Service in a Distributed Fashion to a Mobile Device.” Plaintiff is the assignee of all rights, title, and interest in and to the ’914 patent, including the right to recover damages for present, past, and future infringement. The ’914 patent is valid and enforceable.

12. On December 6, 2016, the United States Patent and Trademark Office issued U.S. Patent No. 9,516,127 (“the ’127 patent”), entitled “Intelligent Alarm Manipulator and Resource Tracker.” Plaintiff is the assignee of all rights, title, and interest in and to the ’127 patent, including the right to recover damages for present, past, and future infringement. The ’127 patent is valid and enforceable.

13. On March 21, 2017, the United States Patent and Trademark Office issued U.S. Patent No. 9,603,056 (“the ’056 patent”), entitled “Mobile Application Traffic Optimization.” Plaintiff is the assignee of all rights, title, and interest in and to the ’056 patent, including the right to recover damages for present, past, and future infringement. The ’056 patent is valid and enforceable.

14. On March 28, 2017, the United States Patent and Trademark Office issued U.S. Patent No. 9,608,968 (“the ’968 patent”), entitled “Connection Architecture for a Mobile Network.” Plaintiff is the assignee of all rights, title, and interest in and to the ’968 patent, including the right to recover damages for present, past, and future infringement. The ’968 patent is valid and enforceable.

15. On May 9, 2017, the United States Patent and Trademark Office issued U.S. Patent No. 9,648,557 (“the ’557 patent”), entitled “System, Method, and Computer-Readable Medium For User Equipment Decision-Making Criteria For Connectivity and Handover,” after full and fair examination. Plaintiff is the assignee of all rights, title, and interest in and to the ’557 patent and possesses all rights of recovery under the ’557 patent, including the right to recover damages for present, past, and future infringement. The ’557 patent is valid and enforceable.

16. On July 18, 2017, the United States Patent and Trademark Office issued U.S. Patent No. 9,712,476 (“the ’476 patent”), entitled “Secure End-To-End Transport Through Intermediary Nodes.” Plaintiff is the assignee of all rights, title, and interest in and and to the ’476 patent and possesses all rights of recovery under the ’476 patent, including the right to recover damages for present, past, and future infringement. The ’476 patent is valid and enforceable.

17. On July 18, 2017, the United States Patent and Trademark Office issued U.S. Patent No. 9,712,986 (“the ’986 patent”), entitled “Mobile Device Configured for Communicating with Another Mobile Device Associated with an Associated User.” Plaintiff is the assignee of all rights, title, and interest in and to the ’986 patent, including the right to recover damages for present, past, and future infringement. The ’986 patent is valid and enforceable.

18. On September 19, 2017, the United States Patent and Trademark Office issued U.S. Patent No. 9,769,176 (“the ’176 patent”), entitled “Multiple Data Store Authentication,” after full and fair examination. Plaintiff is the assignee of all rights, title, and interest in and to the ’176 patent and possesses all rights of recovery under the ’176 patent, including the right to

recover damages for present, past, and future infringement. The ’176 patent is valid and enforceable.

19. On July 17, 2018, the United States Patent and Trademark Office issued U.S. Patent No. 10,027,619 (“the ’619 patent”), entitled “Messaging Centre for Forwarding E-mail.” Plaintiff is the assignee of all rights, title, and interest in and to the ’619 patent, including the right to recover damages for present, past, and future infringement. The ’619 patent is valid and enforceable.

20. On July 31, 2018, the United States Patent and Trademark Office issued U.S. Patent No. 10,039,029 (“the ’029 patent”), entitled “Predictive Fetching of Mobile Application Traffic.” Plaintiff is the assignee of all rights, title, and interest in and to the ’029 patent, including the right to recover damages for present, past, and future infringement. The ’029 patent is valid and enforceable.

21. On October 2, 2018, the United States Patent and Trademark Office issued U.S. Patent No. 10,091,734 (“the ’734 patent”), entitled “Optimizing Mobile Network Traffic Coordination Across Multiple Applications Running On a Mobile Device.” Plaintiff is the assignee of all rights, title, and interest in and to the ’734 patent and possesses all rights of recovery under the ’734 patent, including the right to recover damages for present, past, and future infringement. The ’734 patent is valid and enforceable.

22. On October 23, 2018, the United States Patent and Trademark Office issued U.S. Patent No. 10,110,534 (“the ’534 patent”), entitled “Connection Architecture for a Mobile Network,” after full and fair examination. Plaintiff is the assignee of all rights, title, and interest in and to the ’534 patent and possesses all rights of recovery under the ’534 patent, including the

right to recover damages for present, past, and future infringement. The ’534 patent is valid and enforceable.

23. On November 20, 2018, the United States Patent and Trademark Office issued U.S. Patent No. 10,135,771 (“the ’771 patent”), entitled “Secure End-To-End Transport Through Intermediary Nodes,” after full and fair examination. Plaintiff is the assignee of all rights, title, and interest in and to the ’771 patent and possesses all rights of recovery under the ’771 patent, including the right to recover damages for present, past, and future infringement. The ’771 patent is valid and enforceable.

24. On March 26, 2019, the United States Patent and Trademark Office issued U.S. Patent No. 10,243,962 (“the ’962 patent”), entitled “Multiple Data Store Authentication” after full and fair examination. Plaintiff is the assignee of all rights, title, and interest in and to the ’962 patent and possesses all rights of recovery under the ’962 patent, including the right to recover damages for present, past, and future infringement. The ’962 patent is valid and enforceable.

25. The inventions of the patents-in-suit address technological problems and provide technological solutions and were not well-understood, routine, or conventional at the time of the invention. A person of ordinary skill in the art reading the patents-in-suit and their claims would understand that the patents’ disclosures and claims are drawn to solving specific, technical problems. Moreover, a person of ordinary skill in the art would understand that the claimed subject matter represents advancement in the technical fields of the patents-in-suit. The claims do not preempt all techniques for or approaches to accomplishing the same or a similar end to what they recite. For example, the claims do not preempt the use of the techniques taught in the prior art cited on the face of the patents-in-suit. The large volume of prior art cited on the faces

of the patents-in-suit, none of which, as the Examiners found, discloses or render obvious the claimed inventions further shows that the claims are not well-understood, routine, or conventional.

26. The inventions of the ’539 and ’734 patents resolve technological problems related to “high battery consumption due to the constant need of powering/re-powering the radio module.” ’539, Col. 2:26-29; ’734, Col. 1:65-2:58. Further, with respect to the ’539 patent, the patent resolves technological problems related to the mobile device’s ability to efficiently and flexibly optimize and block data traffic transmission, including through a user selected power save mode and optimization techniques related to the mobile device’s sharing of the same user account with another mobile device. ’539, Col. 2:26-29; 32:3-24. Further, with respect to the ’734 patent, the patent resolves technological problems related to blocking outgoing background application requests, a power-save mode for optimizing data traffic, and an application-by-application user selection technique for optimizing data traffic. ’734, Col. 49:23-56. Because existing solutions did not provide adequate power savings, particularly for small mobile devices constrained by the size of their battery, a need existed for a solution that would better reduce devices’ power consumption.

27. The claims of the ’539 and ’734 patents do not merely recite the performance of some business practice known from the precomputer world along with the requirement to perform it on a computer. Instead, the claims of the ’539 and ’734 patents recite one or more inventive concepts that are rooted in computerized technology, and overcome technical problems specifically arising in that realm. For instance, the patents explain that the “issue [of battery consumption] has been exacerbated by the rapid increase of poplurity of network-initiated functionalities.” ’539, Col. 1:67-2:3; ’734, Col. 2:11-16. The patents identify technical

difficulties in finding a solution to this problem, explaining that prior art "solutions typically assume lack of coordination between the user, the application and the network." '539, Col. 2:11-29; '734, Col. 2:17-43. The patents further explain that application protocols may provide long-lived connections but that requires a "connection remains usable by periodically sending some data, often called a keep-alive message, to the server and making sure the server is receiving the data" and "the cumulative effect of multiple applications performing this individually will amount to small pieces of data being sent very frequently." *Id.* A need existed for an efficient and flexible approach to optimizing traffic on a mobile device by blocking transmission of some traffic and allowing other traffic based on, among other things, the nature of the traffic and user selections. *Id.* The claimed inventive solutions include methods and apparatus that solve this need and claim improved mobile devices, content servers, and methods.

28. A person of ordinary skill in the art reading the '539 and '734 patents would understand that the patents' disclosure and claims are drawn to solving a specific, technical problem to provide improved battery life in mobile devices through optimization, regulation, and maintenance of data transmissions, including through a user-selected power save mode. The claims do not preempt the use of all techniques taught in the field. For example, they do not preempt use of the techniques taught in the prior art cited on the face of the '539 and '734 patents or described in the specification.

29. A person of ordinary skill in the art would understand that the claims of the '539 and '734 patents are directed to a specific improvement for improved battery consumption through optimization, regulation, and maintenance of data transmission, including through specific user-selection techniques. Accordingly, each claim of the '539 and '734 patents recites

a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent claiming an abstract concept.

==========================

30. The inventions of the ’550 patent resolve technological problems related to power consumption in mobile devices that change the frequency of network data requests. ’550, Col. 9:1-24. Because existing solutions resulted in high power consumption, a need existed for a solution that would send application data requests without such a large impact on power consumption. ’550, Col. 1:26-2:7; 9:1-24; p. 163 of 60/403,249; *see also* U.S. Patent Nos. 8,023,475 at Col. 3:46-51 and 5,465,394 at Col. 2:66-3:6. The ’550 patent addresses this need. *Id.*

31. The ’550 patent does not merely recite the performance of some business practice known from the precomputer world along with the requirement to perform it on a computer. Instead, the claims of the ’550 patent recite one or more inventive concepts that are rooted in computerized technology and overcome problems specifically arising in that realm. For example, the claims of the ’550 patent teach specific methods and devices to change the frequency at which application data requests are sent based on the amount of battery power remaining, including through the use of at least two power management modes. These inventive solutions overcome one or more problems of the prior art, including the high power consumption of systems that synchronized frequently. Col. 1:26-2:7; 9:1-24. As detailed by the specification, the prior techniques suffered drawbacks such that a new and novel solution was required. *Id.*

32. A person of ordinary skill in the art reading the ’550 patent and its claims would understand that the patent’s disclosure and claims are drawn to solving a specific, technical problem arising in communication between mobile devices and networks. Moreover, a person of

ordinary skill in the art would understand that the claims' subject matter presents advancements in the field of power management. The claims do not preempt all approaches for improving power management. For example, the claims do not preempt use of the techniques taught in the prior art references cited on the face of the patent.

33. A person of ordinary skill in the art would understand that the claims of the '550 patent are directed to specific improvements in power management in mobile devices that change the frequency of network data requests. Accordingly, each claim of the '550 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent claiming an abstract concept.

==============================

34. The inventions of the '914 patent resolve technological problems related to the distribution of content to multiple devices in an integrated way. '914, Col. 1:59-2:27. Using existing solutions, it was difficult for service providers who were not also mobile network service providers to provide an integrated service offering to their customers to meet the customers' needs. '914, Col. 1:49-55. Because existing solutions did not provide an integrated experience for users, a need existed for a solution that would seamlessly provide content to multiple devices.

35. The '914 patent does not merely recite the performance of some business practice known from the precomputer world along with the requirement to perform it on a computer. Instead, the claims of the '914 patent recite one or more inventive concepts that are rooted in computerized technology, and overcome problems specifically arising in that realm. For example, the claims of the '914 patent teach receiving, on a content selection interface of a first device, an indication of content available from a content provider server, receiving selected

content in response to a user selection of content available from the content provider server, wherein the selected content is automatically transmitted from the content provider server to a second device that is owned by the same owner as the first device, wherein each of the first device and the second device transmit a unique authentication token for authenticating each of the first device and the second device, and wherein the selected content is received at the first device through a first connection and received at the second device through a second connection distinct from the first connection. ’914, Col. 21:35-51. As detailed by the specification, the prior techniques suffered drawbacks such that a new and novel solution was required.

36. A person of ordinary skill in the art reading the ’914 patent and its claims would understand that the patent’s disclosure and claims are drawn to solving a specific, technical problem arising in mobile device provisioning. Moreover, a person of ordinary skill in the art would understand that the claims’ subject matter presents advancements in the field of mobile device provisioning. The claims do not preempt all types of mobile device provisioning. For example, the claims do not preempt use of the techniques taught in the prior art references cited on the face of the patent.

37. A person of ordinary skill in the art would understand that the claims of the ’914 patent are directed to specific improvements in mobile device provisioning. Accordingly, each claim of the ’914 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent claiming an abstract concept.

===================================

38. The inventions of the ’127 patent resolve technological problems related to the performance of periodic or scheduled tasks in mobile applications. ’127, Col. 17:2-15. Indeed, the Court denied a motion filed against the ’127 as allegedly lacking patentable subject matter in

a prior litigation. *See* SEVEN Networks, LLC v. Google, No. 2:17-cv-000442-JRG (EDTX Jan. 17, 2019), Dkt. 607 at 3 (denying Dkt. No. 347); *see also* Dkt. 501 (SEVEN's opposition brief to Dkt. No. 347).

39. Existing solutions had high utilization of network resources, power/battery resources, CPU resources, and memory resources. '127, Col. 17:2-15. Because existing solutions had high power consumption and consumption of other resources, a need existed for a solution that would give users control over background tasks and improve power and resource consumption. The specification of the '127 patent provides such a solution. '127, Col. 4:3-7.

40. The '127 patent does not merely recite the performance of some business practice known from the precomputer world along with the requirement to perform it on a computer. Instead, the claims of the '127 patent recite one or more inventive concepts that are rooted in computerized technology, and overcome problems specifically arising in that realm. For example, certain claims of the '127 patent recite specific methods and devices to receive a selection from a user whether to optimize traffic of a first application executing in a background of the mobile device, to optimize background traffic of the first application, to receive a selection from a user whether to enter a power save mode, and to adjust a timing of activities of a second application upon selection to enter the power save mode to reduce usage of at least one resource of the mobile device. These inventive solutions overcome one or more problems of the prior art. As detailed by the specification, the prior techniques suffered drawbacks such that a new and novel solution was required.

41. A person of ordinary skill in the art reading the '127 patent and its claims would understand that the patent's disclosure and claims are drawn to solving a specific, technical problem arising in communication between mobile devices and networks. Moreover, a person of

ordinary skill in the art would understand that the claims' subject matter presents advancements in the field of mobile devices. The claims do not preempt all types of optimization of background execution in mobile devices. For example, the claims do not preempt use of the techniques taught in the prior art references cited on the face of the patent.

42. In light of the foregoing, a person of ordinary skill in the art would understand that the claims of the '127 patent are directed to specific improvements in optimization of background execution in mobile devices. Accordingly, each claim of the '127 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent claiming an abstract concept.

==============================

43. The inventions of the '056 and '029 patents resolve technological problems related to "high battery consumption due to the constant need of powering/re-powering the radio module." '056, Col. 2:32-36; '029, Col. 2:38-41. They resolve technological problems occurring with modern mobile applications which frequently send and receive data in mobile networks that are designed "for high-throughput of large amounts of data, not for applications that require frequent, but low-throughput and/or small amounts of data." '056, Col. 1:60-2:1; '029, Col. 2:2-7. Because existing solutions assumed lack of coordination between the user, the application and the network, and because existing solutions did not mitigate the cumulative effect of multiple applications' keep-alive traffic, a need existed for a solution that would reduce the need to power a mobile device's radio frequently. '056, Col. 2:18-36; '029, Col. 2:24-41.

44. The '056 and '029 patents do not merely recite the performance of some business practice known from the precomputer world along with the requirement to perform it on a computer. Instead, the claims of the '056 and '029 patents recite one or more inventive concepts

that are rooted in computerized technology, and overcome problems specifically arising in that realm. For example, the claims of the ’029 patent teach specific methods and devices to predict and fetch data for an application, wherein at least some of the fetched data is for background requests made by the application, and wherein a second connection is established. These inventive solutions overcome one or more problems of the prior art. As detailed by the specification, the prior techniques suffered drawbacks such that a new and novel solution was required.

45. A person of ordinary skill in the art reading the ’056 and ’029 patents and their claims would understand that the patents’ disclosure and claims are drawn to solving a specific, technical problem arising in communication between mobile devices and networks. Moreover, a person of ordinary skill in the art would understand that the claims’ subject matter presents advancements in the field of network optimization. The claims do not preempt all types of optimization of mobile networks, or all types of batching or predictive fetching. For example, the claims do not preempt use of the techniques taught in the dozens of prior art references cited on the face of each patent.

46. In light of the foregoing, a person of ordinary skill in the art would understand that the claims of the ’056 and ’029 patents are directed to specific improvements for optimizing network communication and power consumption of mobile devices, including improvements in batching and predictive fetching of application data. Accordingly, each claim of the ’056 and ’029 patents recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent claiming an abstract concept.

===========================

47. The ’534 and ’968 patents address a technological problem and provide a technological solution for improving existing systems that were "cumbersome, do[ ] not operate in real time, and require[ ] sending a large number of email messages over the Internet." ’534 at Col. 1:24-2:5; ’968 at 1:23-65. Because existing solutions did not efficiently provide access to the data users needed, a need existed for a simpler, more stream-lined solution. ’534 at Col. 1:61-63; ’968 at 1:59-61.

48. The ’534 and ’968 patents do not merely recite the performance of some business practice known from the precomputer world along with the requirement to perform it on a computer. Instead, the claims of the ’534 and ’968 patents recite one or more inventive concepts that are rooted in computerized technology, and overcome problems specifically arising in that realm. For example, the ’534 patent describes a server that receives a first connection associated with a first device and a first message over that first connection that comprises a data query for a latest version of data at a second device, generates a second message for a second device based on the first message, receives a second connection associated with the second device and a third message from the second device containing a latest version of data stored at the second device, generates a fourth message containing data from the third message, and sends the fourth message over the first connection. *See, e.g.,* ’534 Col. 9:44-10:7. The ’968 patent further describes a server operable to receive a first connection from a first device, authenticate the first device over the first connection, receive a first transaction from the first device, wherein the first connection is maintained independently of the first transaction, generate a trigger for the second device notifying the second device of new data from the first transaction, and receive a second connection from the second device after the generation of the trigger. *See, e.g.,* ’968 Col. 9:42-10:8. These inventive solutions overcome one or more problems of the prior art, and are

technological solutions to the technological problem identified in the specification. As detailed by the specification, the prior techniques suffered drawbacks such that a new and novel solution was required. With respect to the ’968 patent, unlike a cumbersome, slow architecture, this architecture allows a small notification to be quickly and efficiently pushed to the second device and further allows the second device to obtain larger amounts of data over a second connection after it has received a triggering notification. With respect to the ’534 patent, unlike prior cumbersome, slow architectures, the architecture allows information to be quickly updated between multiple devices efficiently and on demand.

49. A person of ordinary skill in the art reading the ’534 and ’968 patents and their claims would understand that the patents’ disclosure and claims are drawn to solving a specific, technical problem arising in network communication. Moreover, a person of ordinary skill in the art would understand that the claims’ subject matter presents advancements in the field of network communication. The claims do not preempt all types of network communication. For example, the claims do not preempt use of the techniques taught in the prior art references cited on the face of each patent.

50. In light of the foregoing, a person of ordinary skill in the art would understand that the claims of the ’534 and ’968 patents are directed to specific improvements for network communication. Accordingly, each claim of the ’534 and ’968 patents recite a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent claiming an abstract concept.

======================

51. The ’557 patent addresses a technological problem and provides a technological solution for determining how, and under what specific conditions to best switch from a Wi-Fi to

a cellular connection. *See, e.g.,* Col. 1:42-2:5. In prior systems, which would only use one type of connection or which would not make a careful decision of which type of connection to use, poor call quality and/or needlessly high power consumption would result. *See* Col. 2:40-45. For example, in buildings where cellular coverage is poor, yet Wi-Fi coverage is good, Wi-Fi is often the better choice. Col. 2:36-40. On the other hand, Wi-Fi networks can be less reliable for voice calls in some circumstances, so it may be preferable to route voice calls over cellular even when Wi-Fi is available. *See* Col. 9:33-35. Because existing solutions did not effectively optimize for factors like call quality and power consumption, a need existed for a solution that would automatically route communications over the best technology for the circumstances. Col. 1:27-2:54.

52. The '557 patent does not merely recite the performance of some business practice known from the precomputer world along with the requirement to perform it on a computer. Instead, the claims of the '557 patent recite one or more inventive concepts that are rooted in computerized technology, and overcome problems specifically arising in that realm. The '557 claims are directed to a mobile device (and method of operating the same) with a specific set of limitations. The patent describes a technological problem, for example, users "demanding more personalized services and greater flexibility" (Col. 1:44-49) in mixing and matching different types of services. Col. 1:27-61. The patent solves a technological problem, for example, by providing a flexible and efficient approach to switching between services, including by switching to expensive cellular data based on a detected first condition and detected second condition, evaluated user settings, the time responsiveness of the cellular network, and the application executing on the mobile device. The patent provides technological solutions that select the best network to use to communicate. These inventive solutions overcome one or more problems of

the prior art, including poor call quality and high power consumption. As detailed by the specification, the prior techniques suffered drawbacks such that a new and novel solution was required.

53. A person of ordinary skill in the art reading the ’557 patent and its claims would understand that the patent’s disclosure and claims are drawn to solving a specific, technical problem arising in network selection. Moreover, a person of ordinary skill in the art would understand that the claims’ subject matter presents advancements in the field of secure network communications. The claims do not preempt all types of switching from one type of connection to another. For example, the claims do not preempt use of the techniques taught in the prior art references cited on the face of the patent.

54. In light of the foregoing, a person of ordinary skill in the art would understand that the claims of the ’557 patent are directed to specific improvements in network selection that overcome problems in the prior art. Accordingly, each claim of the ’557 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent claiming an abstract concept.

==========================

55. The ’476 patent addresses technological problems related to security and malicious eavesdropping in data communication over a network to another device and provides technological solutions. *See, e.g.,* Col. 1:33-62. As noted in the file history, the ’476 patent teaches inventions with “uniquely distinct features” over the “closest prior arts.” ’476 FH, March 8, 2017 Office Action, at pages 2-3. Because existing solutions had security risks if intermediary servers were compromised, a need existed for a solution that would allow data to be easily transmitted. The ’476 patent provides a simpler, more streamlined solution.

56. The ’476 patent does not merely recite the performance of some business practice known from the precomputer world along with the requirement to perform it on a computer. Instead, the claims of the ’476 patent recite one or more inventive concepts that are rooted in computerized technology, and overcome problems specifically arising in that realm. For example, the patent teaches an improved computing device (and server) that can securely and reliably communicate data to a mobile device using a specific token while reducing the security risk presented by an intermediary server: “The intermediary network processing nodes may be given access to the encryption key used to encrypt the data. However, decrypting the packets at the intermediary points presents a security risk. For example, an eavesdropper may be able to access the data after being decrypted at the intermediary network processing nodes.” Col. 1:41-45. The patent provides technological solutions that provide for security even if an intermediary server is compromised. These inventive solutions overcome one or more problems of the prior art, including security risks. As detailed by the specification, the prior techniques suffered drawbacks such that a new and novel solution was required.

57. A person of ordinary skill in the art reading the ’476 patent and its claims would understand that the patent’s disclosure and claims are drawn to solving a specific, technical problem arising in secure network communications. Moreover, a person of ordinary skill in the art would understand that the claims’ subject matter presents advancements in the field of secure network communications. The claims do not preempt all types of secure network communications. For example, the claims do not preempt use of the techniques taught in the prior art references cited on the face of the patent.

58. In light of the foregoing, a person of ordinary skill in the art would understand that the claims of the ’476 patent are directed to specific improvements in secure network

communications which use specific communication procedures that overcome problems in the prior art. Accordingly, each claim of the ’476 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent claiming an abstract concept.

==========================

59. The inventions of the ’986 patent resolve technological problems related to the distribution of content to multiple devices in an integrated way. ’986, Col. 1:57-2:25. Using existing solutions, it was difficult for service providers who were not also mobile network service providers to provide an integrated service offering to their customers to meet the customers’ needs. ’986, Col. 1:47-53. Because existing solutions did not provide an integrated experience for users, a need existed for a solution that would provide content securely and seamlessly to multiple devices.

60. The ’986 patent does not merely recite the performance of some business practice known from the precomputer world along with the requirement to perform it on a computer. Instead, the claims of the ’986 patent recite one or more inventive concepts that are rooted in computerized technology, and overcome problems specifically arising in that realm. For example, the claims of the ’986 patent teach authenticating the first device of a user, accessing content, and transmitting a representation of the accessed content to an application at a second device associated with the user, wherein the second device is authenticated over a mobile network. ’986, Col. 20:17-41. As detailed by the specification, the prior techniques suffered drawbacks such that a new and novel solution was required.

61. A person of ordinary skill in the art reading the ’986 patent and its claims would understand that the patent’s disclosure and claims are drawn to solving a specific, technical

problem arising in mobile device integration. Moreover, a person of ordinary skill in the art would understand that the claims' subject matter presents advancements in the field of mobile device integration. The claims do not preempt all types of mobile device integration. For example, the claims do not preempt use of the techniques taught in the prior art references cited on the face of the patent.

62. In light of the foregoing, a person of ordinary skill in the art would understand that the claims of the '986 patent are directed to specific improvements in mobile device integration. Accordingly, each claim of the '986 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent claiming an abstract concept.

==========================

63. The '176 patent addresses technological problems related to the unavailability of a secure and "simplified provisioning" protocol for sending content from a data store to a client device. '176, Col. 1:21-5:40. The patent describes technical difficulties associated with provisioning services to devices: "Other common-place factors in the mobile device world make repeated authentication experiences even more inconvenient. The need to access more than one account (e.g., personal and work e-mail, limited time to access an account (e.g, only a few minutes before a lengthy meeting), and the need to have access to new data in near real-time further evidence the difficulties with repeated authentication. Notwithstanding, few users or enterprises are willing to surrender data security (i.e., no user or device authentication wherein anyone could conceivably access the account or data store given knowledge of the existence of that account or data store) for the sake of convenience. As such, there is a need in the art for authentication access to multiple data stores while maintaining certain security precautions

offered by user or device authentication." '176, Col. 1:51-65. Because existing solutions had a cumbersome authentication process and were difficult to use, a need existed for a solution that would allow data to be easily accessed. The specification of the patents provides a simpler, more streamlined solution. *See, e.g.* '176, Col. 2:14-32; 3:20-30; 4:37-59; 19:32-60.

64. The patent does not merely recite the performance of some business practice known from the precomputer world along with the requirement to perform it on a computer. Instead, the claims of the patent recite one or more inventive concepts that are rooted in computerized technology, and overcome problems specifically arising in that realm. For example, the '176 patent provides technological solutions related to an intermediary server that, among other things, sends a first identifier to a client device to present in a subsequent connection with the server, receives registration information for a data store and a request for the client device to receive information from the data store, and provides access to one or more data stores through an intermediary server-issued second identifier, which is associated with the data store a client device's registration information, and configures a service to receive data from the data store on behalf of the client device based on the second identifier. *See, e.g.,* '176, Col. 2:14-32; 3:20-30; 4:37-59; 19:32-60. These inventive solutions overcome one or more problems of the prior art, including difficulties in accessing content and difficulty in authenticating users. As detailed by the specification, the prior techniques suffered drawbacks such that a new and novel solution was required.

65. A person of ordinary skill in the art reading the '176 patent and its claims would understand that the patent's disclosure and claims are drawn to solving a specific, technical problem arising in communication between client devices and networks. Moreover, a person of ordinary skill in the art would understand that the claims' subject matter presents advancements

in the field of messaging. The claims do not preempt all types of messaging. For example, the claims do not preempt use of the techniques taught in the prior art references cited on the face of the patent.

66. In light of the foregoing, a person of ordinary skill in the art would understand that the claims of the ’176 patent are directed to specific improvements in messaging which use specific communication procedures that overcome problems in the prior art. Accordingly, each claim of the ’176 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent claiming an abstract concept.

==========================

67. The inventions of the ’619 patent resolve technological problems related to the difficulty of setting up authentication when forwarding messages between multiple devices. ’619, Col. 1:49-56, 2:4-6. In existing solutions, configuring client software at the mobile terminal was difficult because of user interface restrictions in typical mobile terminals. ’619, Col. 2:4-6. Separate accounts were typically required, each with separate authentication. ’619, Col. 1:50-52. Because existing solutions were cumbersome and difficult to configure, a need existed for a solution that would allow users to set up message forwarding quickly and securely. The specification of the ’619 patent provides such a solution, allowing forwarding to be configured quickly and securely using an optically scanned service activation code and encryption key. ’619, Col. 5:3-12.

68. The ’619 patent does not merely recite the performance of some business practice known from the precomputer world along with the requirement to perform it on a computer. Instead, the claims of the ’619 patent recite one or more inventive concepts that are rooted in computerized technology, and overcome problems specifically arising in that realm. For

example, certain claims of the ’619 patent teach specific methods and devices to optically receive information including a displayed service activation code from a remote device, register the remote device, and receive, encrypt, and send a message. These inventive solutions overcome one or more problems of the prior art. As detailed by the specification, the prior techniques suffered drawbacks such that a new and novel solution was required.

69. A person of ordinary skill in the art reading the ’619 patent and its claims would understand that the patent’s disclosure and claims are drawn to solving a specific, technical problem arising in communication between mobile devices and networks. Moreover, a person of ordinary skill in the art would understand that the claims’ subject matter presents advancements in the field of mobile devices. The claims do not preempt all types of registration or message forwarding in mobile devices. For example, the claims do not preempt use of the techniques taught in the prior art references cited on the face of the patent.

70. In light of the foregoing, a person of ordinary skill in the art would understand that the claims of the ’619 patent are directed to specific improvements in optimization of background execution in mobile devices. Accordingly, each claim of the ’619 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent claiming an abstract concept.

==============================

71. The ’771 patent addresses technological problems related to malicious eavesdropping, security, and efficient communication to a second computer over a mobile network. *See, e.g.,* Col. 1:42-2:5. The claims address a specific problem as related to computer technology when using intermediary network processing nodes for data transactions. Col. 1:42-2:5; ’771 FH, July 5, 2018 Remarks. The specification never suggests that the claimed

combination is well-understood, routine, or conventional, and none of the cited prior art discloses the claim combination of elements. *Id.* The claims as a whole recite "a specific combination of elements that realize an improvement of computer functionality." *Id.* Because existing solutions had security risks if intermediary servers were compromised, a need existed for a solution that would allow data to be securely and easily transmitted. The specification of the '771 patent provides a simpler, more streamlined solution.

72. The '771 patent does not merely recite the performance of some business practice known from the precomputer world along with the requirement to perform it on a computer. Instead, the claims of the '771 patent recite one or more inventive concepts that are rooted in computerized technology, and overcome problems specifically arising in that realm. For example, the patent teaches an improved computing device that can securely and reliably communicate data to a second computer over a mobile network using a token issued by an intermediary server while reducing the security risk presented by an intermediary server: "The intermediary network processing nodes may be given access to the encryption key used to encrypt the data. However, decrypting the packets at the intermediary points presents a security risk. For example, an eavesdropper may be able to access the data after being decrypted at the intermediary network processing nodes." Col. 1:50-55. The patent provides technological solutions that provide for security even if an intermediary server is compromised. These inventive solutions overcome one or more problems of the prior art, including security risks. As detailed by the specification, the prior techniques suffered drawbacks such that a new and novel solution was required.

73. A person of ordinary skill in the art reading the '771 patent and its claims would understand that the patent's disclosure and claims are drawn to solving a specific, technical

problem arising in secure network communications. Moreover, a person of ordinary skill in the art would understand that the claims' subject matter presents advancements in the field of secure network communications. A person of ordinary skill in the art reading the patent would understand that it teaches technological solutions that provide for security even if an intermediary server is compromised. The claims do not preempt all types of secure network communications. For example, the claims do not preempt use of the techniques taught in the prior art references cited on the face of the patent.

74. In light of the foregoing, a person of ordinary skill in the art would understand that the claims of the '771 patent are directed to specific improvements in secure network communications which use specific communication procedures that overcome problems in the prior art. Accordingly, each claim of the '771 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent claiming an abstract concept.

=========================

75. The '962 patent addresses technological problems related to the unavailability of a secure and "simplified provisioning" protocol for sending content from data stores to a client device. Col. 1:31-5:53. The patent describes technical difficulties associated with provisioning services to devices: "Other common-place factors in the mobile device world make repeated authentication experiences even more inconvenient. The need to access more than one account (e.g., personal and work e-mail, limited time to access an account (e.g, only a few minutes before a lengthy meeting), and the need to have access to new data in near real-time further evidence the difficulties with repeated authentication. Notwithstanding, few users or enterprises are willing to surrender data security (i.e., no user or device authentication wherein anyone could conceivably

access the account or data store given knowledge of the existence of that account or data store) for the sake of convenience. As such, there is a need in the art for authentication access to multiple data stores while maintaining certain security precautions offered by user or device authentication." Col. 1:63-2:9. In addition, in previous systems it was difficult to send notifications to devices that could not accept incoming IP connections. Because existing solutions had a cumbersome authentication process and were difficult to use, a need existed for a solution that would allow data to be easily accessed. The specification of the '962 patent provides a simpler, more streamlined solution. '962, Col. 2:25-6:7.

76. The '962 patent does not merely recite the performance of some business practice known from the precomputer world along with the requirement to perform it on a computer. Instead, the claims of the '962 patent recite one or more inventive concepts that are rooted in computerized technology, and overcome problems specifically arising in that realm. For example, the claims of the '962 patent teach specific procedures for communicating between client devices and servers, including sending a second identifier to the client device in response to received registration information, and the use of a keep-alive message to maintain a subsequent IP connection to reduce the need for a device to accept inbound IP connections and allow efficient communication. These inventive solutions overcome one or more problems of the prior art, including difficulties in accessing content, difficulty in authenticating users, and the inability to receive incoming notifications. As detailed by the specification, the prior techniques suffered drawbacks such that a new and novel solution was required. The patent provides technological solutions related to an intermediary server that, among other things, sends a first identifier to a client device to present in a subsequent connection with the server, receives registration information for a data store and a request for the client device to receive information

from the plurality of data stores, and provides access to the data stores through an intermediary server-issued second identifier, which is associated with the data store a client device's registration information, and configures a service to receive data from a data store on behalf of the client device based on the second identifier. *See, e.g.,* Col. 2:25-42; 3:30-40; 4:47-5:3; 21:15-45. The patent also provides for the use of a subsequent IP connection between the client device and the server maintained by keep-alive messages from the client device. *See, e.g.,* Col. 1:31-2:21; Col. 2:54-61; 17:15-24; 19:64-20:6.

77. A person of ordinary skill in the art reading the '962 patent and its claims would understand that the patent's disclosure and claims are drawn to solving a specific, technical problem arising in communication between client devices and networks. Moreover, a person of ordinary skill in the art would understand that the claims' subject matter presents advancements in the field of messaging. The claims do not preempt all types of messaging. For example, the claims do not preempt use of the techniques taught in the prior art references cited on the face of the patent.

78. In light of the foregoing, a person of ordinary skill in the art would understand that the claims of the '962 patent are directed to specific improvements in messaging which use specific communication procedures that overcome problems in the prior art. Accordingly, each claim of the '962 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent claiming an abstract concept.

V. CLAIMS FOR PATENT INFRINGEMENT

79. The allegations provided below are exemplary and without prejudice to Plaintiff's infringement contentions provided pursuant to the Court's scheduling order and local rules. Plaintiff's claim construction contentions regarding the meaning and scope of the claim terms will be provided under the Court's scheduling order and local rules. As detailed below, each

element of at least one claim of each of the patents-in-suit is literally present in the accused products. To the extent that any element is not literally present, each such element is present under the doctrine of equivalents. Plaintiff's analysis below should not be taken as an admission that the preamble of each of the claims is limiting, and plantiff reserves the right to argue that the preamble is not limiting for any of the claims. While publicly available information is cited below, Plaintiff may rely on other forms of evidence to show infringement.

80. The accused products include at least the following products, as well as products with reasonably similar functionality.[1] Identification of the accused products will be provided in Plaintiff's infringement contentions pursuant to the Court's scheduling order and local rules.

- '539 accused products: iPhone 4s, iPhone 5, iPhone 5c, iPhone 5s, iPhone 6, iPhone 6 Plus, iPhone 6s, iPhone 6s Plus, iPhone SE, iPhone 7, iPhone 7 Plus, iPhone 8, iPhone 8 Plus, iPhone X, iPhone XR, iPhone XS, and iPhone XS Max, each with iOS 9 or later versions; Apple servers, including those supporting automatic download and/or automatic update functionality;
- '550, '127, and '734 accused products: iPhone 4s, iPhone 5, iPhone 5c, iPhone 5s, iPhone 6, iPhone 6 Plus, iPhone 6s, iPhone 6s Plus, iPhone SE, iPhone 7, iPhone 7 Plus, iPhone 8, iPhone 8 Plus, iPhone X, iPhone XR, iPhone XS, and iPhone XS Max, each with iOS 9 or later versions;
- '914 accused products: iPhone 4s, iPhone 5, iPhone 5c, iPhone 5s, iPhone 6, iPhone 6 Plus, iPhone 6s, iPhone 6s Plus, iPhone SE, iPhone 7, iPhone 7 Plus, iPhone 8, iPhone 8 Plus, iPhone X, iPhone XR, iPhone XS, iPhone XS Max; iPad,

[1] The accused products include varieties with size differences (such as regular and plus versions) or screen display differences (such as non-Retina display and Retina display versions).

iPad 2, iPad (3rd generation), iPad mini, iPad (4th generation), iPad Air, iPad mini 2, iPad Air 2, iPad mini 3, iPad mini 4, iPad Pro, Mac Pro, iMac Pro, MacBook Pro, iMac, MacBook Air, and Mac Mini, each with iOS 5 and later versions or iTunes 10.3 and later versions; Apple servers, including those supporting automatic download and/or automatic update functionality.

- ’986 accused products: iPhone 4s, iPhone 5, iPhone 5c, iPhone 5s, iPhone 6, iPhone 6 Plus, iPhone 6s, iPhone 6s Plus, iPhone SE, iPhone 7, iPhone 7 Plus, iPhone 8, iPhone 8 Plus, iPhone X, iPhone XR, iPhone XS, iPhone XS Max; iPad, iPad 2, iPad (3rd generation), iPad mini, iPad (4th generation), iPad Air, iPad mini 2, iPad Air 2, iPad mini 3, iPad mini 4, iPad Pro, Mac Pro, iMac Pro, MacBook Pro, iMac, MacBook Air, and Mac Mini, each with iOS 8 and later versions or OS X 10.10 and later versions.
- ’056 and ’029 accused products: iPhone 4, iPhone 4s, iPhone 5, iPhone 5c, iPhone 5s, iPhone 6, iPhone 6 Plus, iPhone 6s, iPhone 6s Plus, iPhone SE, iPhone 7, iPhone 7 Plus, iPhone 8, iPhone 8 Plus, iPhone X, iPhone XR, iPhone XS, iPhone XS Max; iPad, iPad 2, iPad (3rd generation), iPad mini, iPad (4th generation), iPad Air, iPad mini 2, iPad Air 2, iPad mini 3, iPad mini 4, each with iOS 7 and later versions;
- ’968 accused products: Apple servers, including those supporting iCloud and/or iMessage functionality;
- ’557 accused products: iPhone 6s, iPhone 6s Plus, iPhone SE, iPhone 7, iPhone 7 Plus, iPhone 8, iPhone 8 Plus, iPhone X, iPhone XR, iPhone XS, iPhone XS Max, iPad, iPad 2, iPad (3rd generation), iPad mini, iPad (4th generation), iPad Air, iPad

mini 2, iPad Air 2, iPad mini 3, iPad mini 4, and iPad Pro, each with iOS 9 and later versions;[2]

- ’476 accused products: iPhone 3GS, iPhone 4, iPhone 4s, iPhone 5, iPhone 5c, iPhone 5s, iPhone 6, iPhone 6 Plus, iPhone 6s, iPhone 6s Plus, iPhone SE, iPhone 7, iPhone 7 Plus, iPhone 8, iPhone 8 Plus, iPhone X, iPhone XR, iPhone XS, iPhone XS Max; iPad, iPad 2, iPad (3rd generation), iPad mini, iPad (4th generation), iPad Air, iPad mini 2, iPad Air 2, iPad mini 3, iPad mini 4, Mac Pro, iMac Pro, MacBook Pro, iMac, MacBook Air, and Mac Mini, each with iOS 5 and later versions or OS X 10.8 and later versions; Apple Watch[3]; Apple servers, including those supporting Apple Push Notification Service (APNs) functionality;
- ’176 and ’962 accused products: Apple servers, including those supporting Apple Push Notification Service (APNs) functionality.
- ’619 accused products: iPhone 4s, iPhone 5, iPhone 5c, iPhone 5s, iPhone 6, iPhone 6 Plus, iPhone 6s, iPhone 6s Plus, iPhone SE, iPhone 7, iPhone 7 Plus, iPhone 8, iPhone 8 Plus, iPhone X, iPhone XR, iPhone XS, and iPhone XS Max, each with iOS 8 and later versions;
- ’534 accused products: Apple servers, including those supporting Find My Friends and Find My iPhone functionality;
- ’771 accused products: iPhone 3GS, iPhone 4, iPhone 4s, iPhone 5, iPhone 5c, iPhone 5s, iPhone 6, iPhone 6 Plus, iPhone 6s, iPhone 6s Plus, iPhone SE, iPhone 7, iPhone 7 Plus, iPhone 8, iPhone 8 Plus, iPhone X, iPhone XR, iPhone XS,

[2] iPad versions without cellular functionality are not accused for the ’557 patent.

[3] All editions and versions of the Apple Watch are accused, including Series 1, 2, 3, and 4.

iPhone XS Max; iPad, iPad 2, iPad (3rd generation), iPad mini, iPad (4th generation), iPad Air, iPad mini 2, iPad Air 2, iPad mini 3, iPad mini 4, Mac Pro, iMac Pro, MacBook Pro, iMac, MacBook Air, and Mac Mini, each with iOS 5 and later versions or OS X 10.8 and later versions; Apple Watch[4]; Apple's servers, including those supporting iCloud functionality.

81. In accordance with 35 U.S.C. § 287, Apple has had actual notice and knowledge of all of the patents-in-suit no later than the filing of this complaint and/or the date this complaint was served upon Apple. On information and belief, Apple continues without license to make, use, import/expert into/from, market, offer for sale, and/or sell in the United States products that infringe the patents-in-suit.

82. Apple has directly and indirectly infringed and continues to directly and indirectly infringe each of the patents-in-suit by engaging in acts constituting infringement under 35 U.S.C. § 271(a) (b), and/or (c).

83. On information and belief, Apple makes, uses, sells, and/or offers to sell accused products and/or components thereof in this district and elsewhere in the United States.

84. On information and belief, Apple imports accused products and/or components into the United States.

85. Apple instructs its customers to use the accused products in manners that infringe the patents-in-suit. For example, Apple provides instruction manuals for the accused products and describes, markets, and/or advertises infringing functionality on its website, at Apple developer conferences, and in other Apple documentation.

[4] All editions and versions of the Apple Watch are accused, including Series 1, 2, 3, and 4.

86. On information and belief, Apple tests each of the accused products in the United States, therebyinfringing the patents-in-suit. On information and belief, Apple uses each of the accused products, including the accused Apple servers, in this district, thus infringing the patents-in-suit.

87. Apple's acts of infringement have caused damage to the Plaintiff. The Plaintiff is entitled to recover from Apple the damages sustained by the Plantiff as a result of Apple's wrongful acts in an amount subject to proof at trial.

88. In the interest of providing detailed averments of infringement, the Plaintiff has identified below at least one claim per patent to demonstrate infringement. However, the selection of claims should not be considered limiting, and additional claims of the patents-in-suit that are infringed by Apple will be disclosed in compliance with the Court's rules related to infringement contentions.

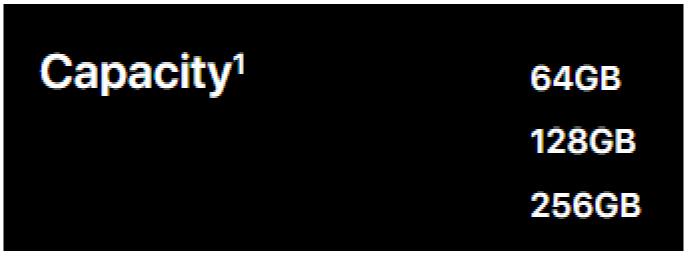
COUNT I: PATENT INFRINGEMENT OF THE '539 PATENT

89. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

90. Apple infringes (literally and/or under the Doctrine of Equivalents) the '539 patent by making, using, offering for sale, selling and/or importing into the United States products and/or methods covered by one or more claims of the '539 patent including at least the '539 accused products noted above. In addition, Apple derives revenue from the activities relating to the '539 accused products.

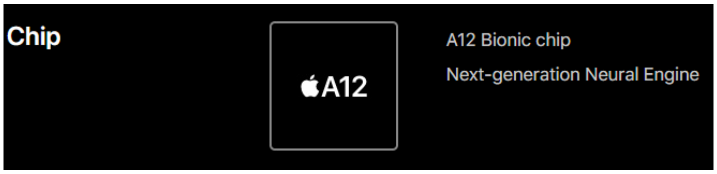
91. For example and as shown below, the '539 accused products infringe at least claim 7 of the '539 patent. For example, and to the extent the preamble is limiting, each of the '539 accused products are "mobile device comprising: a radio; a memory; and a processor

configured for:" As shown below by the exemplary evidence, the '539 accused products satisfy a mobile device comprising a radio, a memory, and a processor.



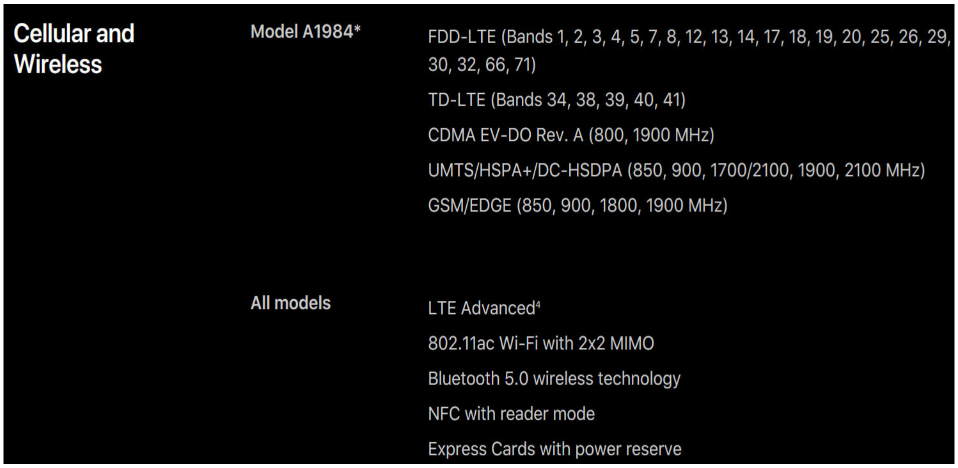
Capacity[1]	64GB 128GB 256GB

https://www.apple.com/iphone-xr/specs/



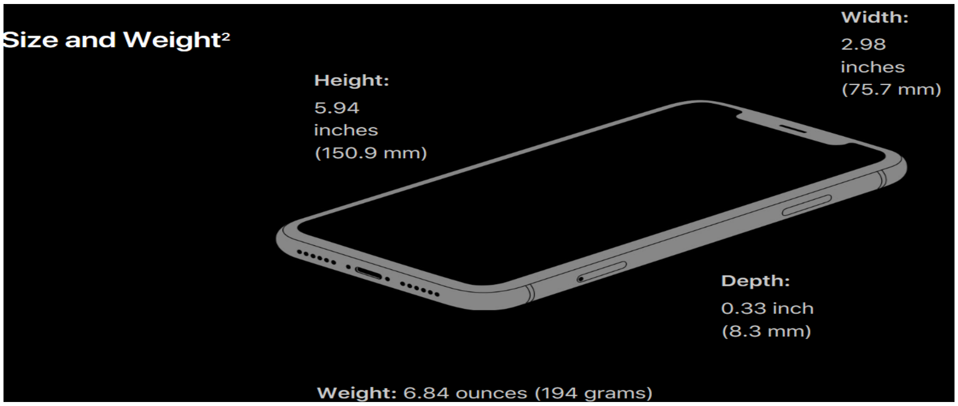
Chip	A12	A12 Bionic chip Next-generation Neural Engine

https://www.apple.com/iphone-xr/specs/



Cellular and Wireless	Model A1984*	FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41) CDMA EV-DO Rev. A (800, 1900 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
	All models	LTE Advanced[4] 802.11ac Wi-Fi with 2x2 MIMO Bluetooth 5.0 wireless technology NFC with reader mode Express Cards with power reserve

https://www.apple.com/iphone-xr/specs/



Size and Weight[2]

Width: 2.98 inches (75.7 mm)

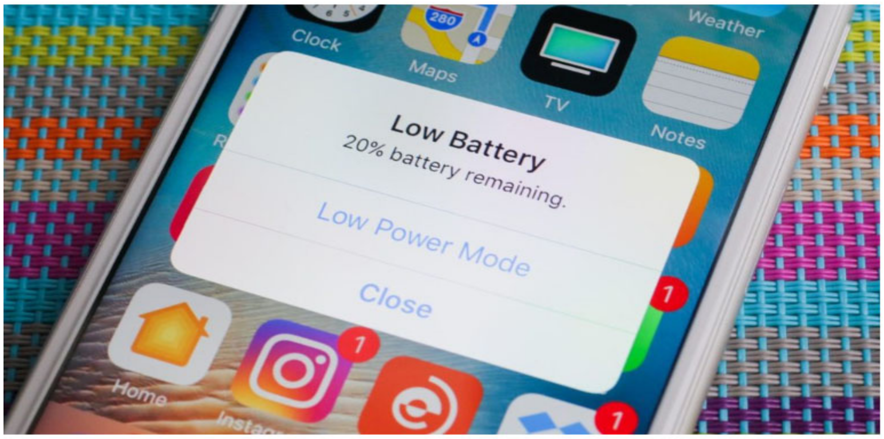
Height: 5.94 inches (150.9 mm)

Depth: 0.33 inch (8.3 mm)

Weight: 6.84 ounces (194 grams)

https://www.apple.com/iphone-xr/specs/

92. Claim 7 of the ’539 patent recites “querying a user by displaying a notification on a user interface of the mobile device to select whether to enter a power save mode.” The ’539 accused products satisfy this limitation. For example, the ’539 accused products query a user by displaying a notification on a user interface of the mobile device to select whether to enter a power save mode, as shown in the exemplary documentation below.

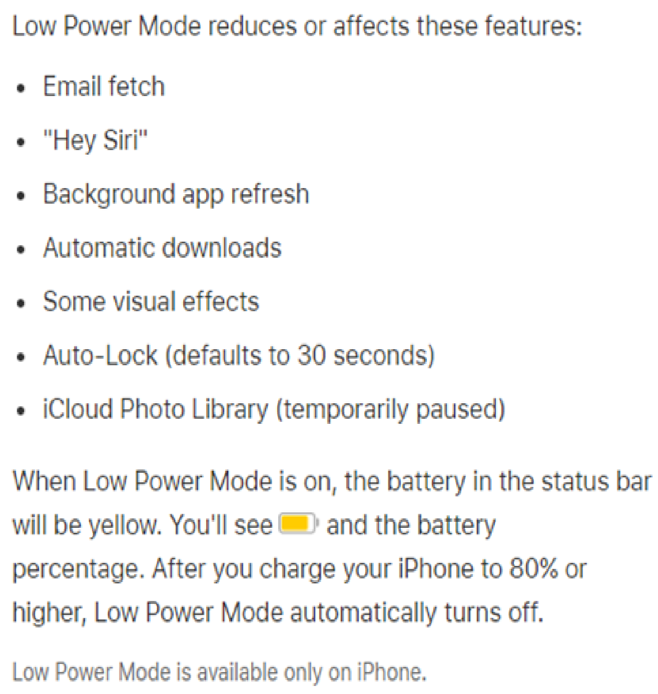


Low Battery

20% battery remaining.

https://www.thefonestuff.com/blog/how-to-improve-iphone-x-battery-life/

93. Claim 7 of the ’539 patent recites “upon selection by a user of entering the power save mode for the mobile device, optimizing traffic at the mobile device by blocking transmission of at least some traffic from the mobile device.” The ’539 accused products satisfy this limitation. For example, in low power mode, automatic downloads are disabled, as shown in the exemplary documentation below.



Low Power Mode reduces or affects these features:

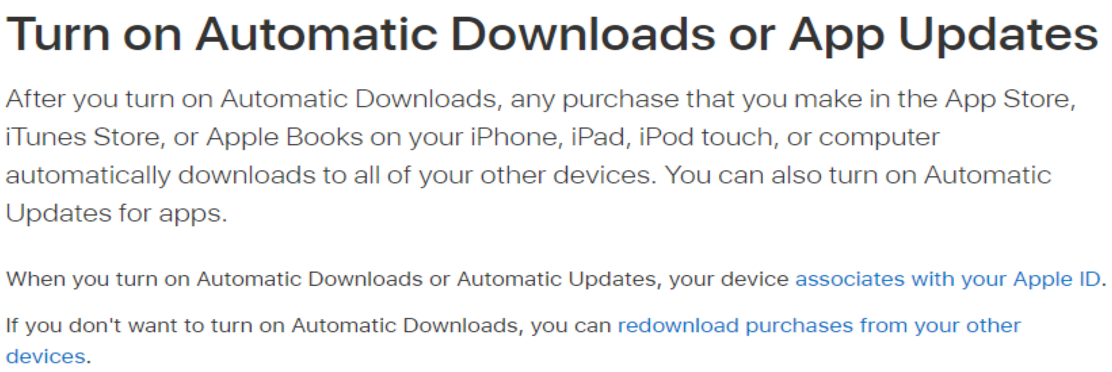
- Email fetch
- "Hey Siri"
- Background app refresh
- Automatic downloads
- Some visual effects
- Auto-Lock (defaults to 30 seconds)
- iCloud Photo Library (temporarily paused)

When Low Power Mode is on, the battery in the status bar will be yellow. You'll see 🔋 and the battery percentage. After you charge your iPhone to 80% or higher, Low Power Mode automatically turns off.

Low Power Mode is available only on iPhone.

https://support.apple.com/en-us/HT205234

94. Claim 7 of the ’539 patent recites “wherein another mobile device shares a same user account with the mobile device.” The ’539 accused products satisfy this limitation. For example, the ’539 accused products operate in an environment in which another mobile device, such as an iPad or Mac computer, is associated with the same Apple ID, as shown in the exemplary documentation below.



Turn on Automatic Downloads or App Updates

After you turn on Automatic Downloads, any purchase that you make in the App Store, iTunes Store, or Apple Books on your iPhone, iPad, iPod touch, or computer automatically downloads to all of your other devices. You can also turn on Automatic Updates for apps.

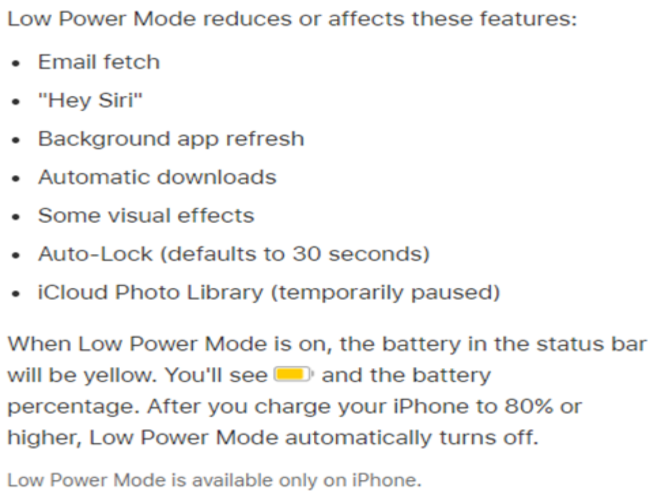
When you turn on Automatic Downloads or Automatic Updates, your device associates with your Apple ID.

If you don't want to turn on Automatic Downloads, you can redownload purchases from your other devices.

https://support.apple.com/en-us/HT202180

95. Claim 7 of the ’539 patent recites “wherein content selected for download at the another mobile device from a server is delayed for download at the mobile device when the

mobile device is in the power save mode." The '539 accused products satisfy this limitation. For example, in low power mode, automatic downloads are disabled such that content purchased on the iPad or Mac is not downloaded to the iPhone while the iPhone is in low power mode, as shown in the exemplary documentation below.



Low Power Mode reduces or affects these features:

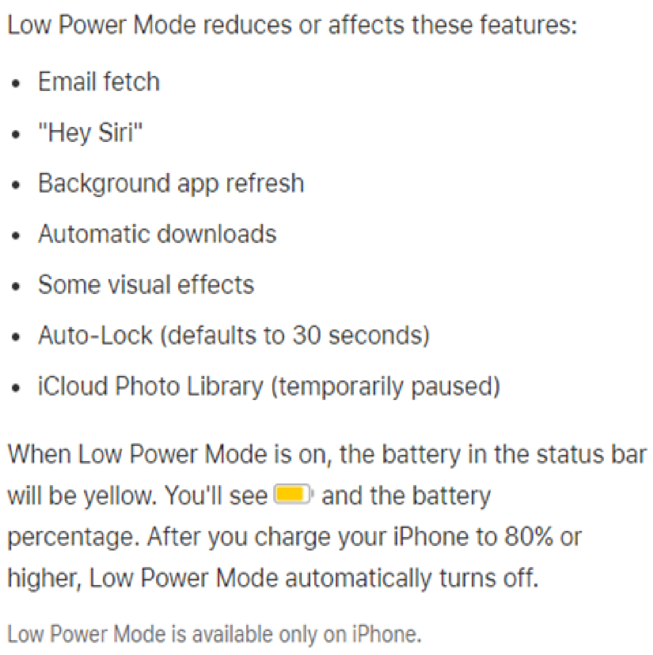
- Email fetch
- "Hey Siri"
- Background app refresh
- Automatic downloads
- Some visual effects
- Auto-Lock (defaults to 30 seconds)
- iCloud Photo Library (temporarily paused)

When Low Power Mode is on, the battery in the status bar will be yellow. You'll see 🟨 and the battery percentage. After you charge your iPhone to 80% or higher, Low Power Mode automatically turns off.

Low Power Mode is available only on iPhone.

https://support.apple.com/en-us/HT205234

96. Claim 7 of the '539 patent recites "wherein the content selected for download at the another mobile device is downloaded at the second mobile device when the second mobile device is not in the power save mode." The '539 accused products satisfy this limitation. For example, when the low power mode is turned off, the automatic downloads will no longer be disabled and the content will be automatically downloaded to the iPhone, as shown in the exemplary documentation below.



Low Power Mode reduces or affects these features:

- Email fetch
- "Hey Siri"
- Background app refresh
- Automatic downloads
- Some visual effects
- Auto-Lock (defaults to 30 seconds)
- iCloud Photo Library (temporarily paused)

When Low Power Mode is on, the battery in the status bar will be yellow. You'll see 🔋 and the battery percentage. After you charge your iPhone to 80% or higher, Low Power Mode automatically turns off.

Low Power Mode is available only on iPhone.

https://support.apple.com/en-us/HT205234

97. Thus, as illustrated above, the ’539 accused products directly infringe one or more claims of the ’539 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the ’539 patent.

98. Apple indirectly infringes the ’539 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple’s customers and end-users, in this district and elsewhere in the United States. For example, Apple’s customers and end-users directly infringe through their use of the inventions claimed in the ’539 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’539 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the ’539 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation,

marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products' capability in an infringing manner, including through the low power mode. *See, e.g.,* https://support.apple.com/en-us/HT205234. As a result of Apple's inducement, Apple's customers and end-users use the Apple products in the way Apple intends and directly infringe the '539 patent. Apple performs these affirmative acts with knowledge of the '539 patent and with the intent, or willful blindness, that the induced acts directly infringe the '539 patent. Apple has had knowledge and notice of the '539 patent at least as of the filing of this complaint.

99. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

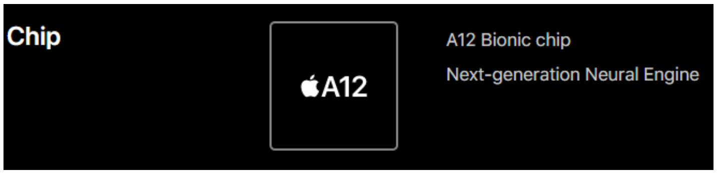
COUNT II: PATENT INFRINGEMENT OF THE '550 PATENT

100. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

101. Apple infringes (literally and/or under the Doctrine of Equivalents) the '550 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the '550 patent including at least the '550 accused products noted above. In addition, Apple derives revenue from the activities relating to the '550 accused products.

102. For example and as shown below, the '550 accused products infringe at least claim 15 of the '550 patent. For example, and to the extent the preamble is limiting, the '550

accused products satisfy a "mobile device located in a mobile network, comprising: a battery; a processor configured to allow the mobile device to:" As shown below by the exemplary evidence, the '550 accused products satisfy a mobile device located in a mobile network that comprises a battery and a processor.



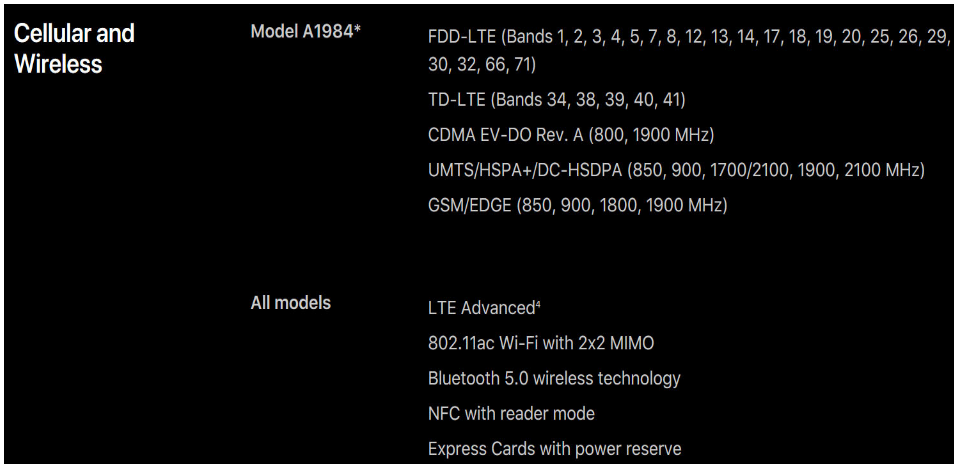
Chip

A12

A12 Bionic chip

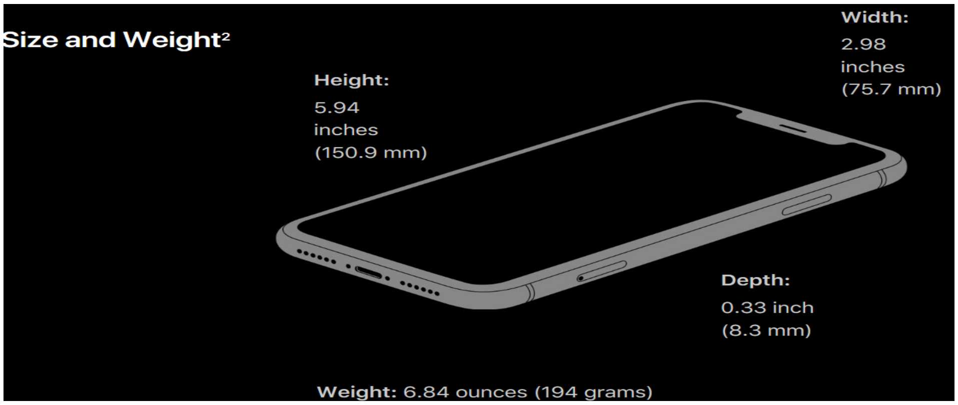
Next-generation Neural Engine

https://www.apple.com/iphone-xr/specs/



Cellular and Wireless	Model A1984*	FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41) CDMA EV-DO Rev. A (800, 1900 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
	All models	LTE Advanced[4] 802.11ac Wi-Fi with 2x2 MIMO Bluetooth 5.0 wireless technology NFC with reader mode Express Cards with power reserve

https://www.apple.com/iphone-xr/specs/



Size and Weight[2]

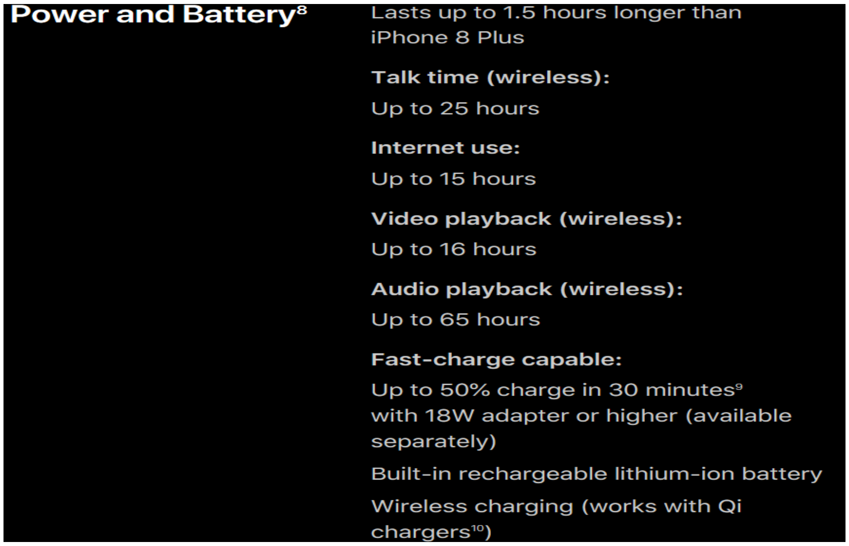
Width:
2.98
inches
(75.7 mm)

Height:
5.94
inches
(150.9 mm)

Depth:
0.33 inch
(8.3 mm)

Weight: 6.84 ounces (194 grams)

https://www.apple.com/iphone-xr/specs/



Power and Battery[8]

Lasts up to 1.5 hours longer than iPhone 8 Plus

Talk time (wireless):
Up to 25 hours

Internet use:
Up to 15 hours

Video playback (wireless):
Up to 16 hours

Audio playback (wireless):
Up to 65 hours

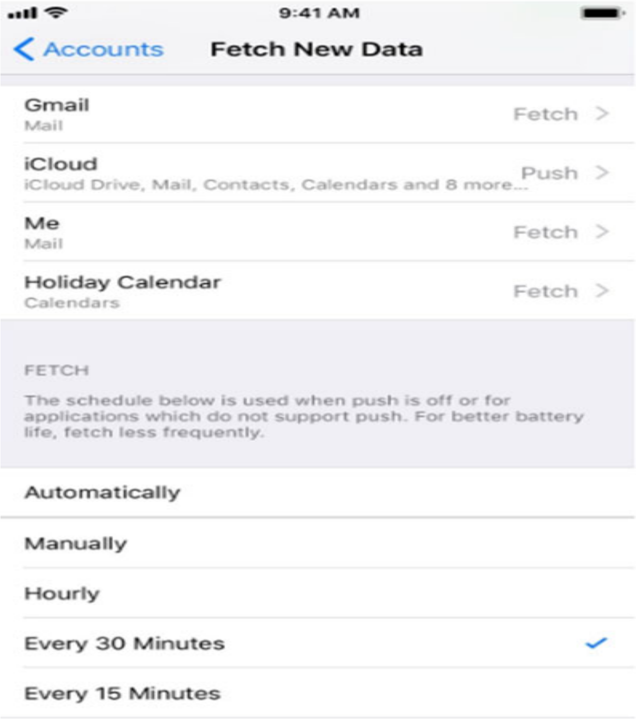
Fast-charge capable:
Up to 50% charge in 30 minutes[9] with 18W adapter or higher (available separately)

Built-in rechargeable lithium-ion battery

Wireless charging (works with Qi chargers[10])

https://www.apple.com/iphone-xr/specs/

103. Claim 15 of the '550 patent recites "send application data requests to a host over a first connection at first frequency." The '550 accused products satisfy this limitation, as shown by the screenshots and documentation below.



Fetch New Data

Gmail Mail	Fetch >
iCloud iCloud Drive, Mail, Contacts, Calendars and 8 more...	Push >
Me Mail	Fetch >
Holiday Calendar Calendars	Fetch >

FETCH

The schedule below is used when push is off or for applications which do not support push. For better battery life, fetch less frequently.

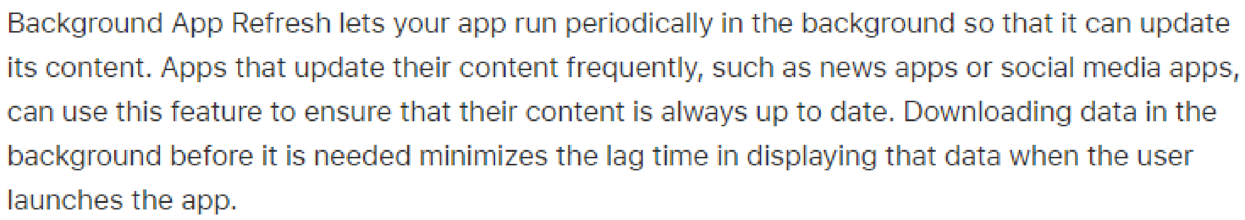
Automatically

Manually

Hourly

Every 30 Minutes ✓

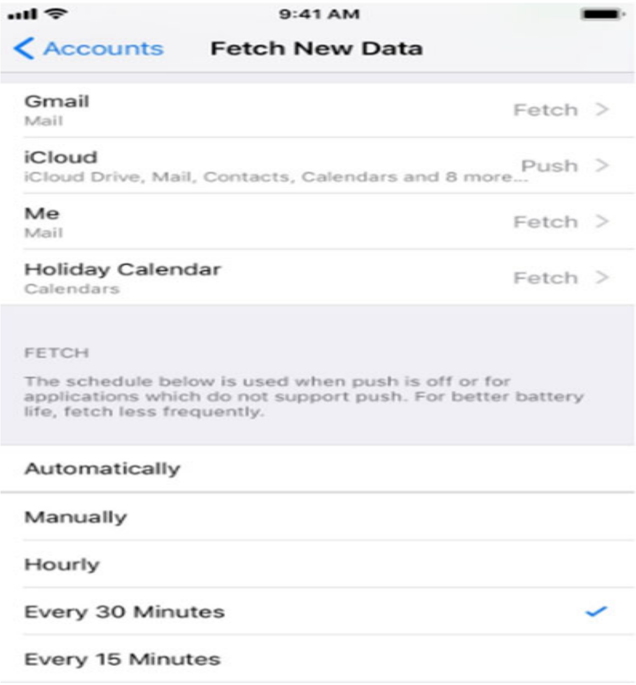
Every 15 Minutes



Background App Refresh lets your app run periodically in the background so that it can update its content. Apps that update their content frequently, such as news apps or social media apps, can use this feature to ensure that their content is always up to date. Downloading data in the background before it is needed minimizes the lag time in displaying that data when the user launches the app.

https://developer.apple.com/documentation/uikit/core_app/managing_your_app_s_life_cycle/preparing_your_app_to_run_in_the_background/updating_your_app_with_background_app_refresh

104. Claim 15 of the '550 patent recites "receive data from the network responsive to the sent application requests." The '550 accused products satisfy this limitation. For example, the '550 accused products receive data in response to application requests.



9:41 AM

< Accounts Fetch New Data

Gmail — Mail — Fetch >

iCloud — iCloud Drive, Mail, Contacts, Calendars and 8 more... — Push >

Me — Mail — Fetch >

Holiday Calendar — Calendars — Fetch >

FETCH

The schedule below is used when push is off or for applications which do not support push. For better battery life, fetch less frequently.

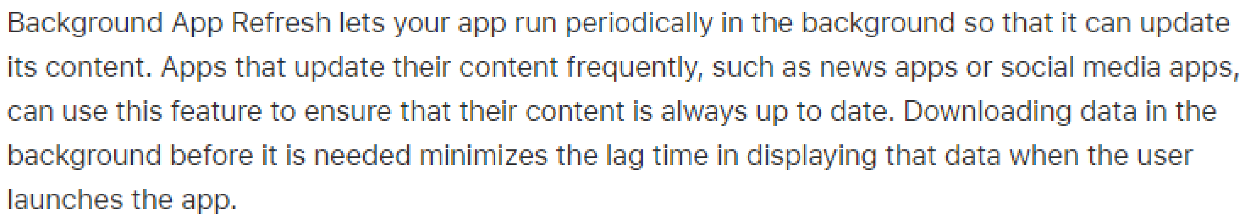
Automatically

Manually

Hourly

Every 30 Minutes ✓

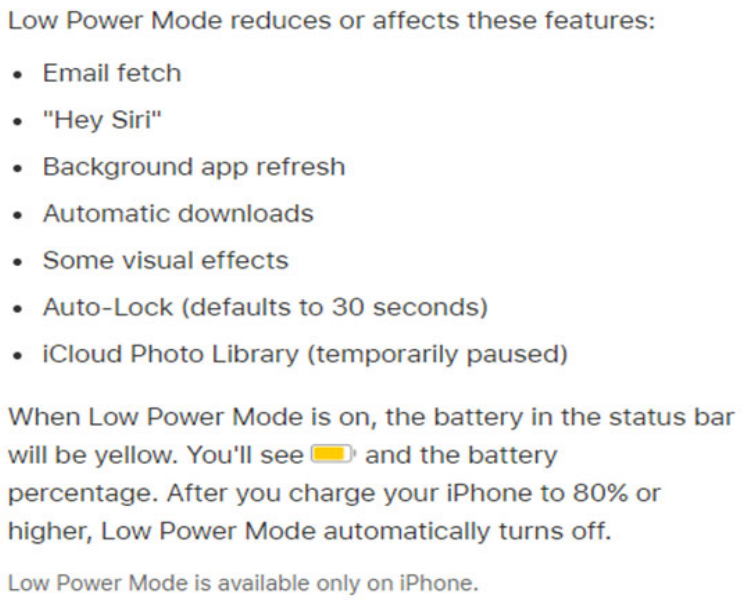
Every 15 Minutes



Background App Refresh lets your app run periodically in the background so that it can update its content. Apps that update their content frequently, such as news apps or social media apps, can use this feature to ensure that their content is always up to date. Downloading data in the background before it is needed minimizes the lag time in displaying that data when the user launches the app.

https://developer.apple.com/documentation/uikit/core_app/managing_your_app_s_life_cycle/preparing_your_app_to_run_in_the_background/updating_your_app_with_background_app_refresh

105. Claim 15 of the ’550 patent recites “select a power management mode from a plurality of power management modes based on an amount of battery power remaining on the mobile device, wherein the selection of a power management mode is further based on the amount of battery power remaining being below a predetermined amount.” The ’550 accused products satisfy this limitation, as shown in the exemplary documentation below.



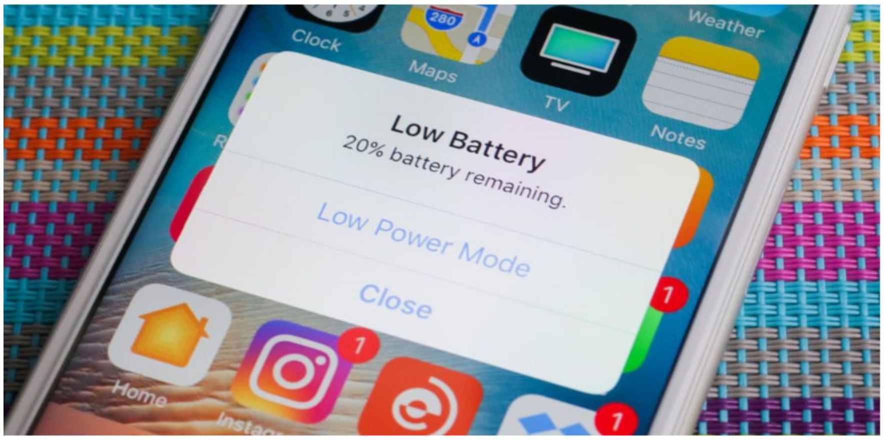
Low Power Mode reduces or affects these features:

- Email fetch
- "Hey Siri"
- Background app refresh
- Automatic downloads
- Some visual effects
- Auto-Lock (defaults to 30 seconds)
- iCloud Photo Library (temporarily paused)

When Low Power Mode is on, the battery in the status bar will be yellow. You'll see 🔋 and the battery percentage. After you charge your iPhone to 80% or higher, Low Power Mode automatically turns off.

Low Power Mode is available only on iPhone.

https://support.apple.com/en-us/HT205234

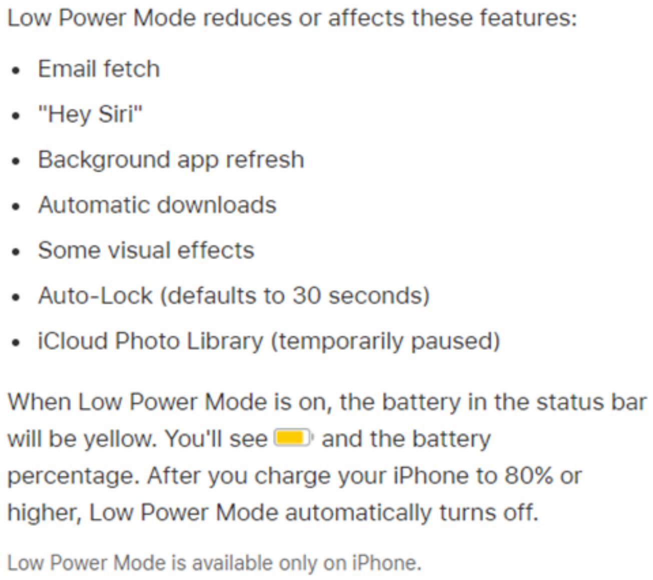


Low Battery
20% battery remaining.
Low Power Mode
Close

https://www.thefonestuff.com/blog/how-to-improve-iphone-x-battery-life/

106. Claim 15 of the ’550 patent recites “change the frequency that application data requests are sent from the first frequency to a second frequency associated with the selected

power management mode." The '550 accused products satisfy this limitation. For example, in different modes, the '550 accused products change the frequency of application data requests, as shown in the exemplary documentation below.



Low Power Mode reduces or affects these features:

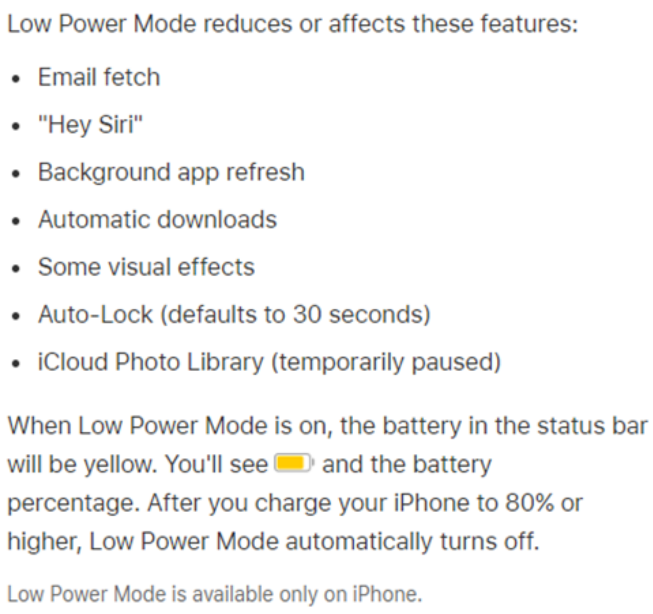
- Email fetch
- "Hey Siri"
- Background app refresh
- Automatic downloads
- Some visual effects
- Auto-Lock (defaults to 30 seconds)
- iCloud Photo Library (temporarily paused)

When Low Power Mode is on, the battery in the status bar will be yellow. You'll see 🔋 and the battery percentage. After you charge your iPhone to 80% or higher, Low Power Mode automatically turns off.

Low Power Mode is available only on iPhone.

https://support.apple.com/en-us/HT205234

107. Claim 15 of the '550 patent recites "wherein at least two of the power management modes are a low power mode configured to conserve the amount of battery power remaining on the mobile device and a normal operation mode." The '550 accused products satisfy this limitation. For example, the '550 accused products support Low Power Mode and normal operation, as shown in the exemplary documentation below.



Low Power Mode reduces or affects these features:

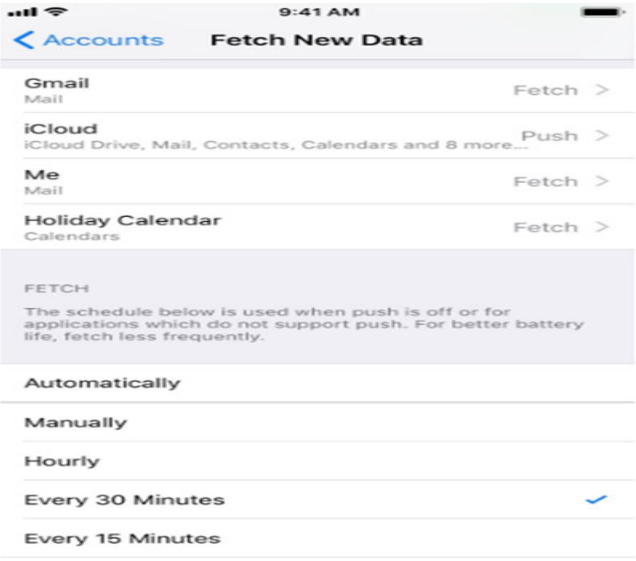
- Email fetch
- "Hey Siri"
- Background app refresh
- Automatic downloads
- Some visual effects
- Auto-Lock (defaults to 30 seconds)
- iCloud Photo Library (temporarily paused)

When Low Power Mode is on, the battery in the status bar will be yellow. You'll see 🟨 and the battery percentage. After you charge your iPhone to 80% or higher, Low Power Mode automatically turns off.

Low Power Mode is available only on iPhone.

https://support.apple.com/en-us/HT205234

108. Claim 15 of the '550 patent recites "wherein the normal operation mode is configured to allow the mobile device to send application data requests more frequently than when the mobile device is in the low power mode." The '550 accused products satisfy this limitation. For example, the '550 accused products send requests more frequently during normal operation, as shown in the exemplary documentation below.



9:41 AM

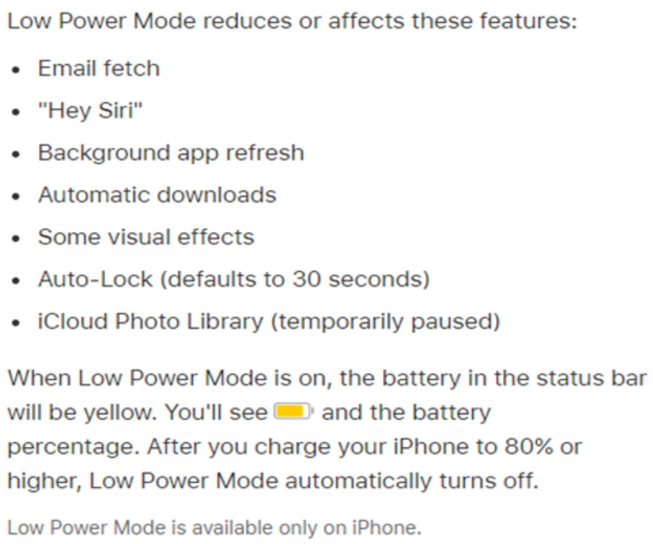
< Accounts **Fetch New Data**

Gmail Mail	Fetch >
iCloud iCloud Drive, Mail, Contacts, Calendars and 8 more...	Push >
Me Mail	Fetch >
Holiday Calendar Calendars	Fetch >

FETCH

The schedule below is used when push is off or for applications which do not support push. For better battery life, fetch less frequently.

Automatically	
Manually	
Hourly	
Every 30 Minutes	✓
Every 15 Minutes	



Low Power Mode reduces or affects these features:

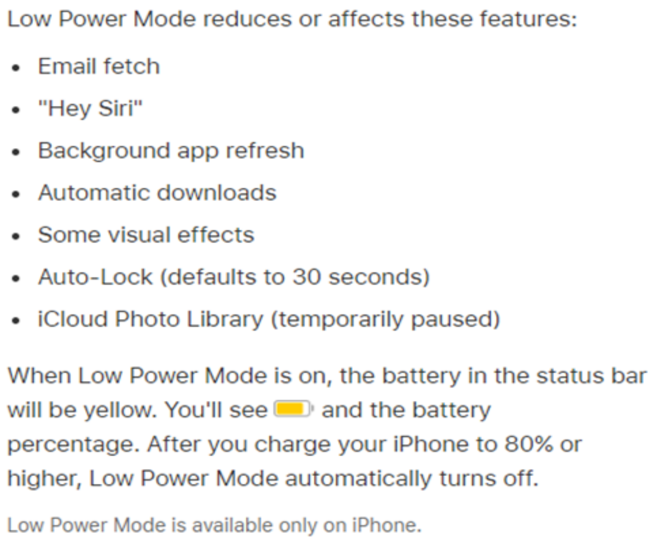
- Email fetch
- "Hey Siri"
- Background app refresh
- Automatic downloads
- Some visual effects
- Auto-Lock (defaults to 30 seconds)
- iCloud Photo Library (temporarily paused)

When Low Power Mode is on, the battery in the status bar will be yellow. You'll see 🔋 and the battery percentage. After you charge your iPhone to 80% or higher, Low Power Mode automatically turns off.

Low Power Mode is available only on iPhone.

https://support.apple.com/en-us/HT205234

109. Claim 15 of the ’550 patent recites “exit the low power mode when an amount of battery remaining is above a predetermined amount.” The ’550 accused products satisfy this limitation. For example, the ’550 accused products exit low power mode when an amount of battery remaining is above a predetermined amount.



Low Power Mode reduces or affects these features:

- Email fetch
- "Hey Siri"
- Background app refresh
- Automatic downloads
- Some visual effects
- Auto-Lock (defaults to 30 seconds)
- iCloud Photo Library (temporarily paused)

When Low Power Mode is on, the battery in the status bar will be yellow. You'll see 🔋 and the battery percentage. After you charge your iPhone to 80% or higher, Low Power Mode automatically turns off.

Low Power Mode is available only on iPhone.

https://support.apple.com/en-us/HT205234

110. Thus, as illustrated above, the ’550 accused products directly infringe one or more claims of the ’550 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in

this district and/or elsewhere in the United States, these devices and thus directly infringes the ’550 patent.

111. Apple indirectly infringes the ’550 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple’s customers and end-users, in this district and elsewhere in the United States. For example, Apple’s customers and end-users directly infringe through their use of the inventions claimed in the ’550 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’550 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the ’550 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products’ capability in an infringing manner, including through the low power mode. *See, e.g.,* https://support.apple.com/en-us/HT205234. As a result of Apple’s inducement, Apple’s customers and end-users use the Apple products in the way Apple intends and directly infringe the ’550 patent. Apple performs these affirmative acts with knowledge of the ’550 patent and with the intent, or willful blindness, that the induced acts directly infringe the ’550 patent. Apple has had knowledge and notice of the ’550 patent at least as of the filing of this complaint.

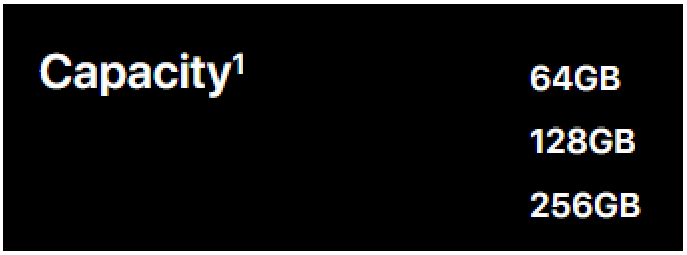
112. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

COUNT III: PATENT INFRINGEMENT OF THE ’914 PATENT

113. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

114. Apple infringes (literally and/or under the Doctrine of Equivalents) the ’914 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the ’914 patent including at least the ’914 accused products noted above. In addition, Apple derives revenue from the activities relating to the ’914 accused products.

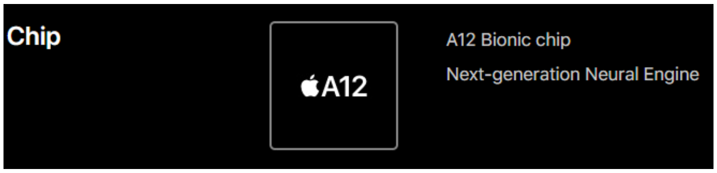
115. For example and as shown below, the ’914 accused products infringe at least claim 21 of the ’914 patent. For example, and to the extent the preamble is limiting, each of the ’914 accused products satisfy a “first device having a memory and a processor, the first device configured for:” As shown below by the exemplary evidence, the ’914 accused products have a memory and a processor.



Capacity[1]

64GB
128GB
256GB

https://www.apple.com/iphone-xr/specs/

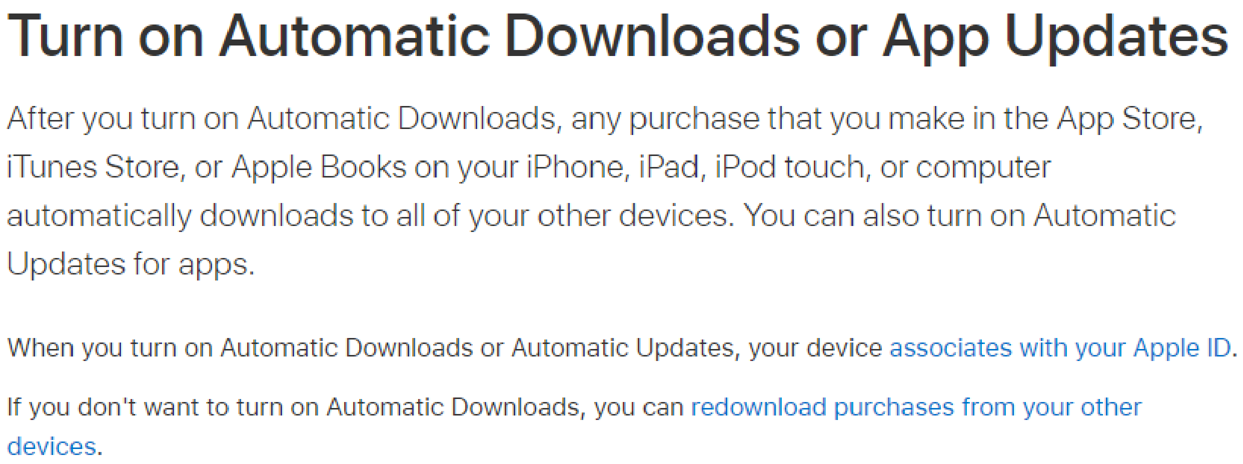


Chip

A12 Bionic chip
Next-generation Neural Engine

https://www.apple.com/iphone-xr/specs/

116. Claim 21 of the ’914 patent recites “receiving, on a content selection interface of a first device, an indication of content available from a content provider server.” The ’914 accused products satisfy this limitation. For example, the ’914 accused products include a content selection interface for receiving an indictation of content available from a content provider server, such as by way of example, content related to the iTunes Store, App Store, Apple Books, etc., as shown by the exemplary documentation below.



Turn on Automatic Downloads or App Updates

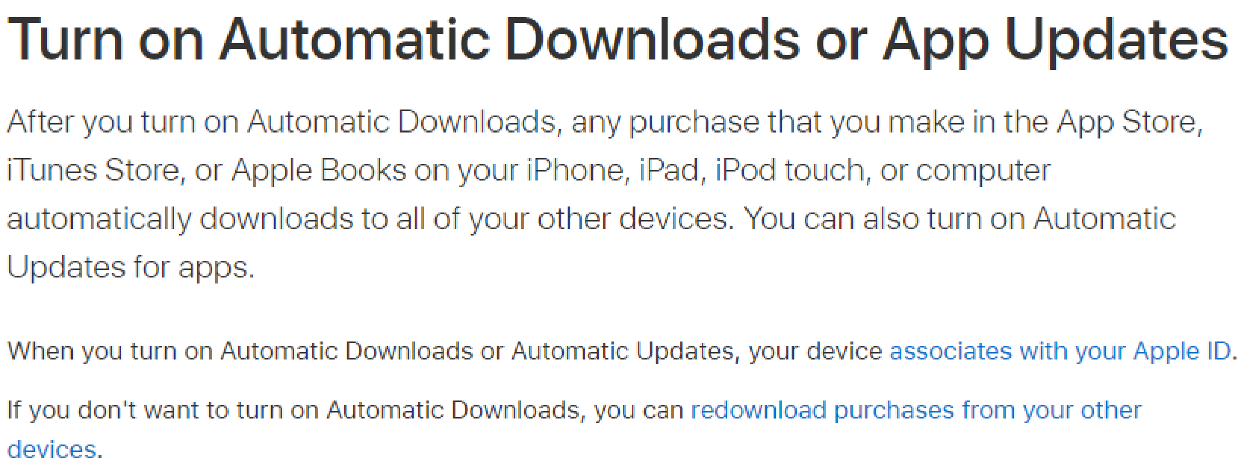
After you turn on Automatic Downloads, any purchase that you make in the App Store, iTunes Store, or Apple Books on your iPhone, iPad, iPod touch, or computer automatically downloads to all of your other devices. You can also turn on Automatic Updates for apps.

When you turn on Automatic Downloads or Automatic Updates, your device associates with your Apple ID.

If you don't want to turn on Automatic Downloads, you can redownload purchases from your other devices.

https://support.apple.com/en-us/HT202180

117. Claim 21 of the ’914 patent recites “receiving selected content in response to a user selection of content available from the content provider server.” The ’914 accused products satisfy this limitation. For example, the ’914 accused products download content from the iTunes Store, App Store, and/or Books in response to user selection, as shown by the exemplary documentation below.



Turn on Automatic Downloads or App Updates

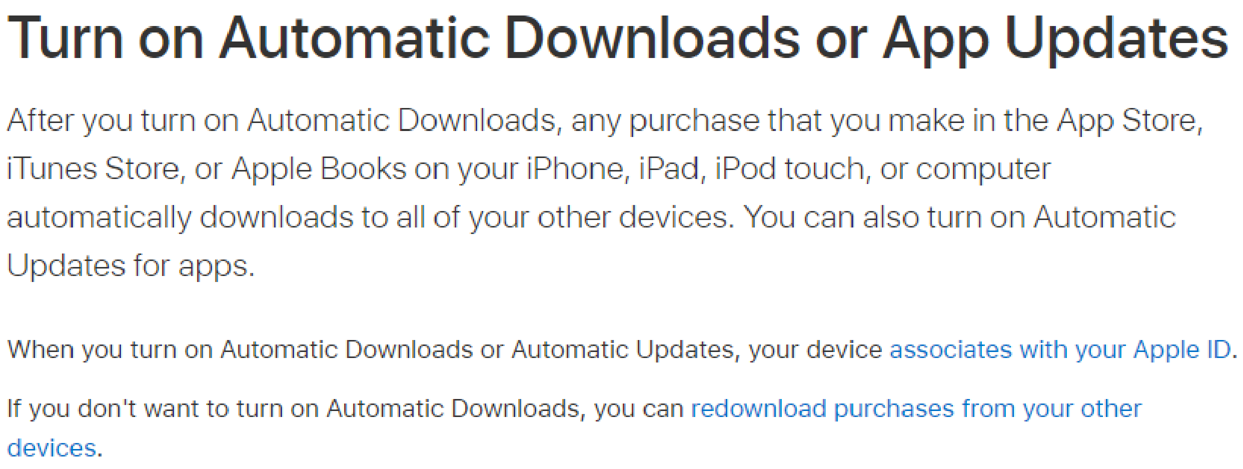
After you turn on Automatic Downloads, any purchase that you make in the App Store, iTunes Store, or Apple Books on your iPhone, iPad, iPod touch, or computer automatically downloads to all of your other devices. You can also turn on Automatic Updates for apps.

When you turn on Automatic Downloads or Automatic Updates, your device associates with your Apple ID.

If you don't want to turn on Automatic Downloads, you can redownload purchases from your other devices.

https://support.apple.com/en-us/HT202180

118. Claim 21 of the ’914 patent recites “wherein the selected content is automatically transmitted from the content provider server to a second device that is owned by the same owner as the first device.” The ’914 accused products satisfy this limitation. For example, through automatic downloads, any purchase in App Store, iTunes, and/or Books is automatically downloaded to all other devices with the same Apple ID, as shown by the exemplary documentation below.



Turn on Automatic Downloads or App Updates

After you turn on Automatic Downloads, any purchase that you make in the App Store, iTunes Store, or Apple Books on your iPhone, iPad, iPod touch, or computer automatically downloads to all of your other devices. You can also turn on Automatic Updates for apps.

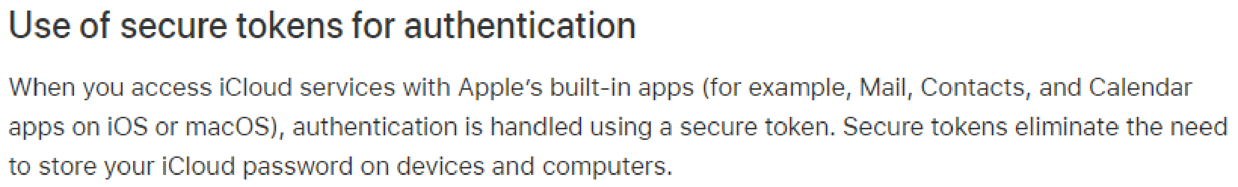
When you turn on Automatic Downloads or Automatic Updates, your device associates with your Apple ID.

If you don't want to turn on Automatic Downloads, you can redownload purchases from your other devices.

https://support.apple.com/en-us/HT202180

119. Claim 21 of the ’914 patent recites “wherein each of the first device and the second device transmit a unique authentication token for authenticating each of the first device

and the second device." The '914 accused products satisfy this limitation. For example, Apple devices use tokens for authentication, as shown by the exemplary documentation below.

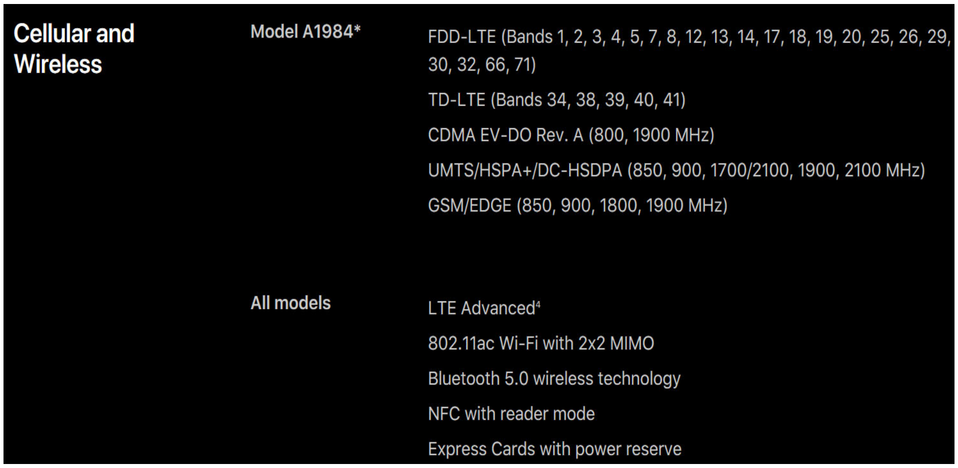


Use of secure tokens for authentication

When you access iCloud services with Apple's built-in apps (for example, Mail, Contacts, and Calendar apps on iOS or macOS), authentication is handled using a secure token. Secure tokens eliminate the need to store your iCloud password on devices and computers.

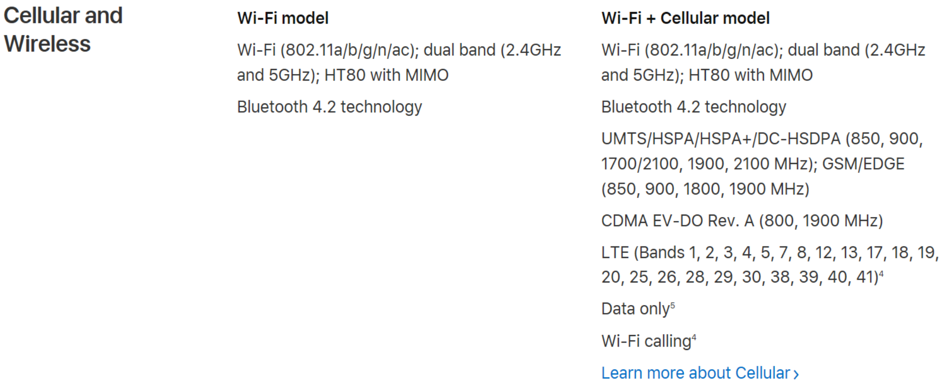
https://support.apple.com/en-us/HT202303

120. Claim 21 of the '914 patent recites "wherein the selected content is received at the first device through a first connection and received at the second device through a second connection distinct from the first connection." The '914 accused products satisfy this limitation. For example, Apple devices are connected by separate connections. For example, an iPhone could be connected by cellular and an iPad connected by Wi-Fi or a separate cellular connection.



Cellular and Wireless	Model A1984*	FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41) CDMA EV-DO Rev. A (800, 1900 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
	All models	LTE Advanced[4] 802.11ac Wi-Fi with 2x2 MIMO Bluetooth 5.0 wireless technology NFC with reader mode Express Cards with power reserve

https://www.apple.com/iphone-xr/specs/



	Wi-Fi model	Wi-Fi + Cellular model
Cellular and Wireless	Wi-Fi (802.11a/b/g/n/ac); dual band (2.4GHz and 5GHz); HT80 with MIMO Bluetooth 4.2 technology	Wi-Fi (802.11a/b/g/n/ac); dual band (2.4GHz and 5GHz); HT80 with MIMO Bluetooth 4.2 technology UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz); GSM/EDGE (850, 900, 1800, 1900 MHz) CDMA EV-DO Rev. A (800, 1900 MHz) LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 28, 29, 30, 38, 39, 40, 41)[4] Data only[5] Wi-Fi calling[4] Learn more about Cellular ›

https://www.apple.com/ipad-9.7/specs/

121. Thus, as illustrated above, the ’914 accused products directly infringe one or more claims of the ’914 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the ’914 patent.

122. Apple indirectly infringes the ’914 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple’s customers and end-users, in this district and elsewhere in the United States. For example, Apple’s customers and end-users directly infringe through their use of the inventions claimed in the ’914 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’914 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the ’914 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products’ capability in an infringing manner, including through the use of automatic downloads and other relevant functionality cited above. *See, e.g.,* https://support.apple.com/en-us/HT202180. As a result of Apple’s inducement, Apple’s customers and end-users use the Apple products in the way Apple intends and directly infringe the ’914 patent. Apple performs these affirmative acts with knowledge of the ’914

patent and with the intent, or willful blindness, that the induced acts directly infringe the ’914 patent. Apple has had knowledge and notice of the ’914 patent at least as of the filing of this complaint.

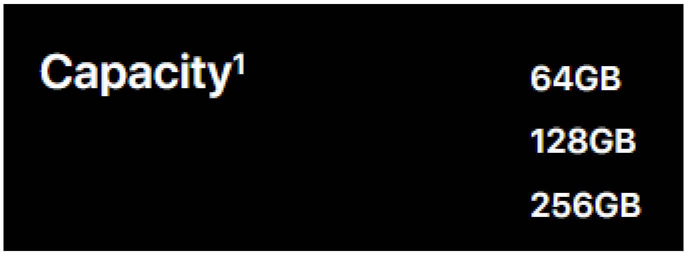
123. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

COUNT IV: PATENT INFRINGEMENT OF THE ’127 PATENT

124. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

125. Apple infringes (literally and/or under the Doctrine of Equivalents) the ’127 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the ’127 patent including at least the ’127 accused products noted above. In addition, Apple derives revenue from the activities relating to the ’127 accused products.

126. For example and as shown below, the ’127 accused products infringe at least claim 33 of the ’127 patent. For example, and to the extent the preamble is limiting, the ’127 accused products satisfy a “mobile device comprising: a memory; a processor in communication with the memory and configured to execute instruction stored in the memory to:” As shown below by the exemplary evidence, the ’127 accused products satisfy mobile devices comprising a memory and a processor.



Capacity[1]

64GB

128GB

256GB

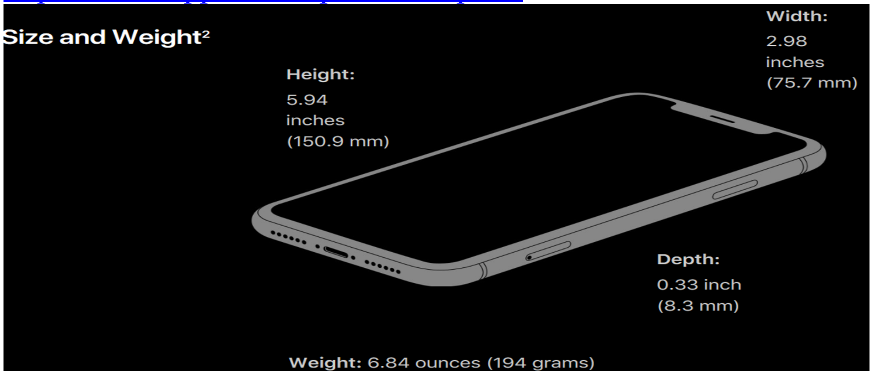
https://www.apple.com/iphone-xr/specs/

Chip

A12 Bionic chip

Next-generation Neural Engine

https://www.apple.com/iphone-xr/specs/



Size and Weight[2]

Height: 5.94 inches (150.9 mm)

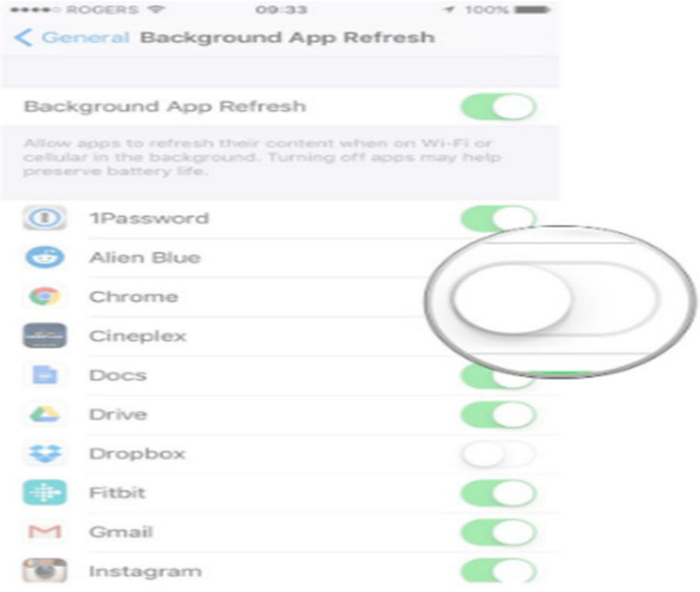
Width: 2.98 inches (75.7 mm)

Depth: 0.33 inch (8.3 mm)

Weight: 6.84 ounces (194 grams)

https://www.apple.com/iphone-xr/specs/

127. Claim 33 of the ’127 patent recites “receive a selection from a user whether to optimize traffic of a first application executing in a background of the mobile device.” The ’127 accused products satisfy this limitation. For example, the ’127 accused products allow, for example, a user to disable background app refresh on an app-by-app basis as shown in the exemplary screenshots below.

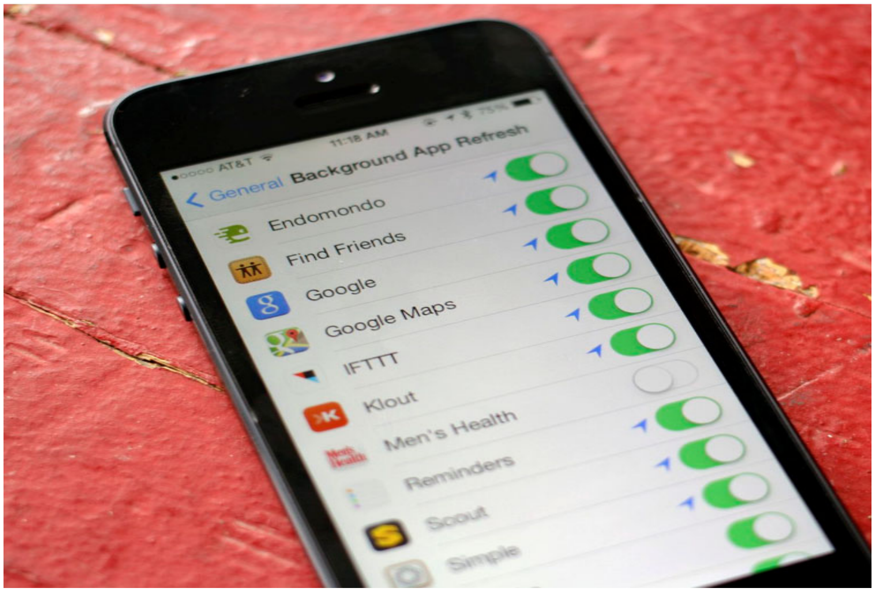


Background App Refresh

Background App Refresh ●

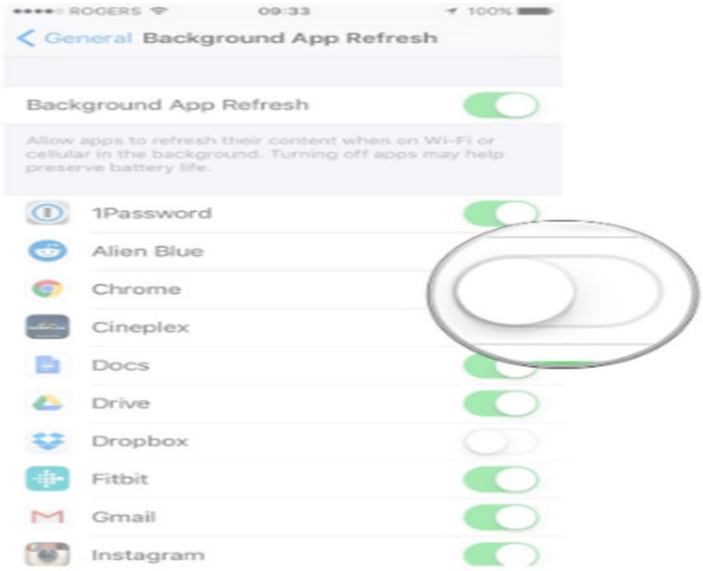
Allow apps to refresh their content when on Wi-Fi or cellular in the background. Turning off apps may help preserve battery life.

1Password ●
Alien Blue
Chrome
Cineplex
Docs ●
Drive ●
Dropbox ○
Fitbit ●
Gmail ●
Instagram ●



https://www.cnet.com/how-to/how-to-turn-off-background-app-refresh-in-ios-7/

128. Claim 33 of the '127 patent recites "optimize background traffic of the first application." The '127 accused products satisfy this limitation. For example, if the user disables background refresh for an app, the '127 accused products will not allow background traffic which conserves network and mobile device resources.

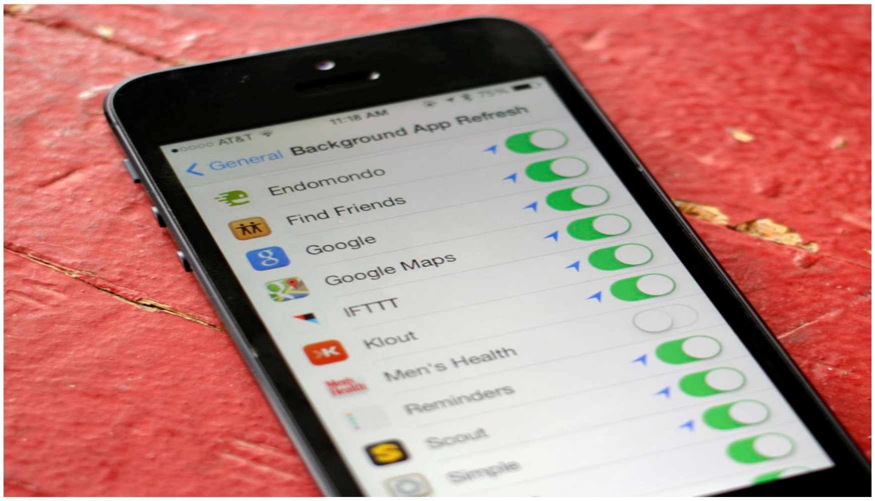

ROGERS 09:33 100%

General Background App Refresh

Background App Refresh

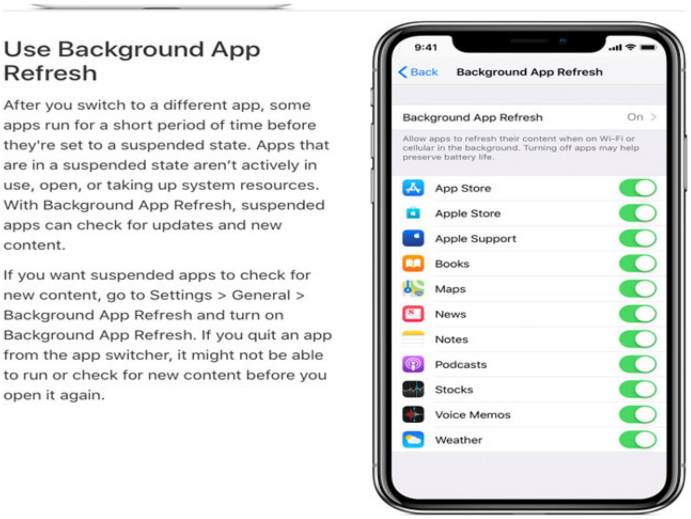
Allow apps to refresh their content when on Wi-Fi or cellular in the background. Turning off apps may help preserve battery life.

1Password
Alien Blue
Chrome
Cineplex
Docs
Drive
Dropbox
Fitbit
Gmail
Instagram



Endomondo
Find Friends
Google
Google Maps
IFTTT
Klout
Men's Health
Reminders
Scout
Simple

https://www.cnet.com/how-to/how-to-turn-off-background-app-refresh-in-ios-7/



Use Background App Refresh

After you switch to a different app, some apps run for a short period of time before they're set to a suspended state. Apps that are in a suspended state aren't actively in use, open, or taking up system resources. With Background App Refresh, suspended apps can check for updates and new content.

If you want suspended apps to check for new content, go to Settings > General > Background App Refresh and turn on Background App Refresh. If you quit an app from the app switcher, it might not be able to run or check for new content before you open it again.

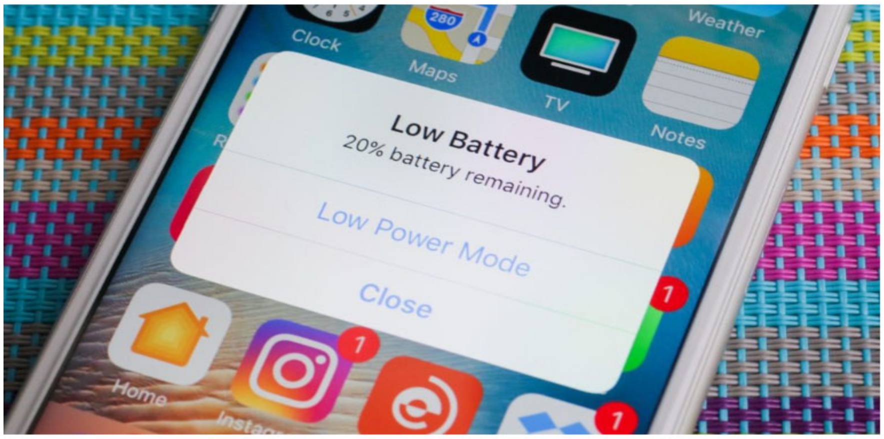
Background App Refresh On >

Allow apps to refresh their content when on Wi-Fi or cellular in the background. Turning off apps may help preserve battery life.

App Store
Apple Store
Apple Support
Books
Maps
News
Notes
Podcasts
Stocks
Voice Memos
Weather

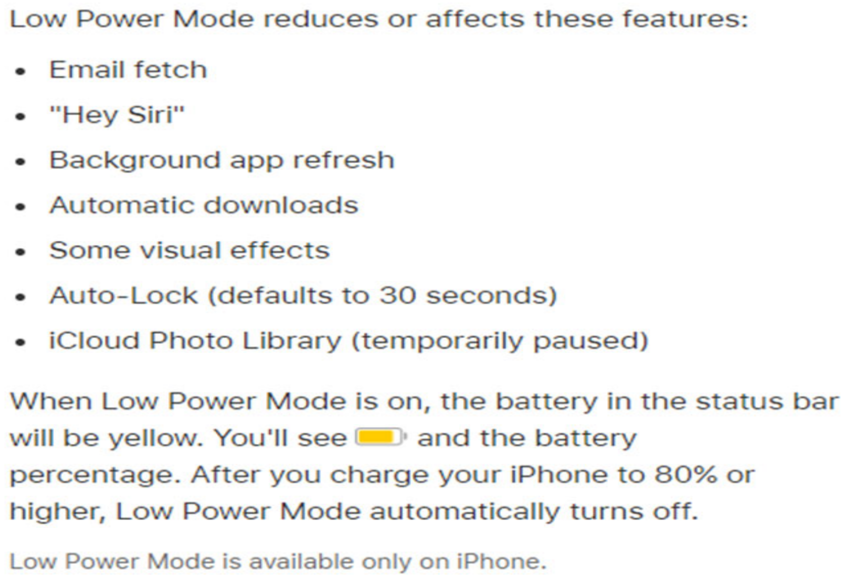
https://support.apple.com/en-us/HT202070

129. Claim 33 of the '127 patent recites "receive a selection from a user whether to enter a power save mode, where the power save mode is based on a battery level of the mobile device." The '127 accused products satisfy this limitation. For example, the user selects whether to enter low power mode when the battery is at 20%.



https://www.thefonestuff.com/blog/how-to-improve-iphone-x-battery-life/

130. Claim 33 of the ’127 patent recites “upon selection to enter the power save mode, adjust a timing of activities of a second application executing in the background of the mobile device to reduce usage of at least one resource of the mobile device.” The ’127 accused products satisfy this limitation. For example, the ’127 accused products adjust a timing of email fetch and background app refresh for other apps during low power mode.



Low Power Mode reduces or affects these features:

- Email fetch
- "Hey Siri"
- Background app refresh
- Automatic downloads
- Some visual effects
- Auto-Lock (defaults to 30 seconds)
- iCloud Photo Library (temporarily paused)

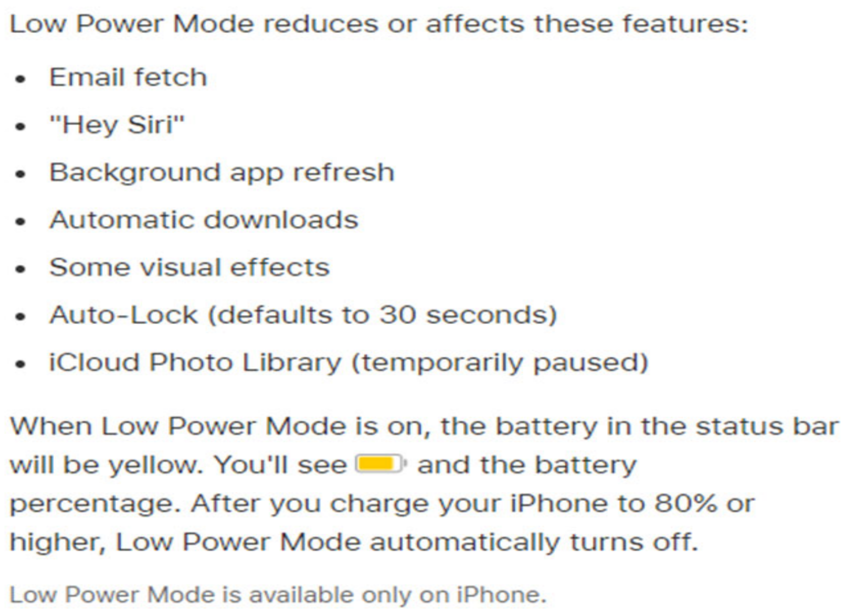
When Low Power Mode is on, the battery in the status bar will be yellow. You'll see 🟨 and the battery percentage. After you charge your iPhone to 80% or higher, Low Power Mode automatically turns off.

Low Power Mode is available only on iPhone.

https://support.apple.com/en-us/HT205234

131. Claim 33 of the ’127 patent recites “exit the power save mode wherein the power save mode is exited based on a battery level or in response to the user directing the mobile device

to exit the power save mode." The '127 accused products satisfy this limitation. For example, the '127 accused products exit low power mode when the battery reaches 80% or when the user turns it off.



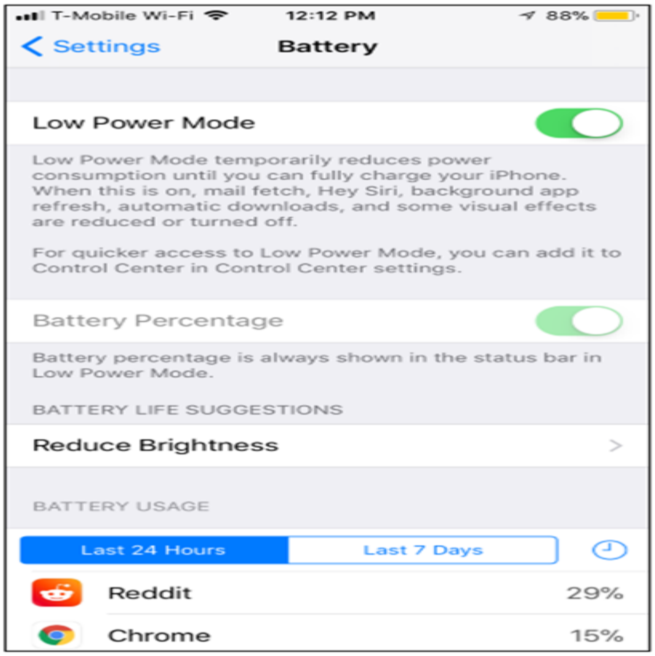
Low Power Mode reduces or affects these features:

- Email fetch
- "Hey Siri"
- Background app refresh
- Automatic downloads
- Some visual effects
- Auto-Lock (defaults to 30 seconds)
- iCloud Photo Library (temporarily paused)

When Low Power Mode is on, the battery in the status bar will be yellow. You'll see 🔋 and the battery percentage. After you charge your iPhone to 80% or higher, Low Power Mode automatically turns off.

Low Power Mode is available only on iPhone.

https://support.apple.com/en-us/HT205234



Low Power Mode: on

Low Power Mode temporarily reduces power consumption until you can fully charge your iPhone. When this is on, mail fetch, Hey Siri, background app refresh, automatic downloads, and some visual effects are reduced or turned off.

For quicker access to Low Power Mode, you can add it to Control Center in Control Center settings.

Battery Percentage: on

Battery percentage is always shown in the status bar in Low Power Mode.

BATTERY LIFE SUGGESTIONS

Reduce Brightness

BATTERY USAGE

Reddit	29%
Chrome	15%

132. Thus, as illustrated above, the ’127 accused products directly infringe one or more claims of the ’127 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the ’127 patent.

133. Apple indirectly infringes the ’127 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple’s customers and end-users, in this district and elsewhere in the United States. For example, Apple’s customers and end-users directly infringe through their use of the inventions claimed in the ’127 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’127 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the ’127 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products’ capability in an infringing manner, including through the use of the low power mode. *See, e.g.,* https://support.apple.com/en-us/HT205234. As a result of Apple’s inducement, Apple’s customers and end-users use the Apple products in the way Apple intends and directly infringe the ’127 patent. Apple performs these affirmative acts with knowledge of the ’127 patent and with the intent, or willful blindness,

that the induced acts directly infringe the ’127 patent. Apple has had knowledge and notice of the ’127 patent at least as of the filing of this complaint.

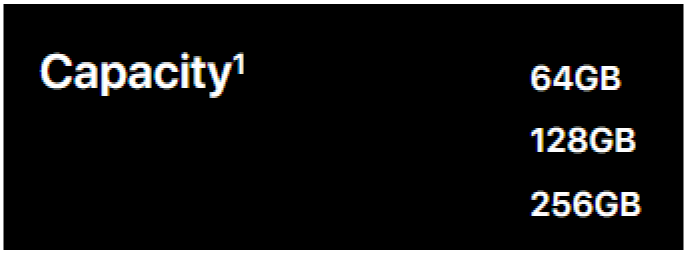
134. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

COUNT V: PATENT INFRINGEMENT OF THE ’056 PATENT

135. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

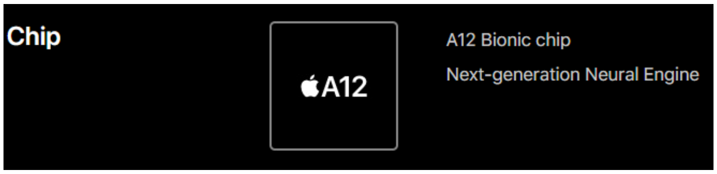
136. Apple infringes (literally and/or under the Doctrine of Equivalents) the ’056 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the ’056 patent including at least the ’056 accused products noted above. In addition, Apple derives revenue from the activities relating to the ’056 accused products.

137. For example and as shown below, the ’056 accused products infringe at least claim 1 of the ’056 patent. For example, and to the extent the preamble is limiting, the ’056 accused products satisfy a “mobile device configured to optimize connections made by the mobile device in a wireless network, the mobile device comprising: a memory; a radio; and a processor, the mobile device configured to:” As shown below by the exemplary evidence, the ’056 accused products satisfy mobile devices configured to optimize connections made by a mobile device in a wireless network, the mobile device comprising a memory, a radio, and a processor.



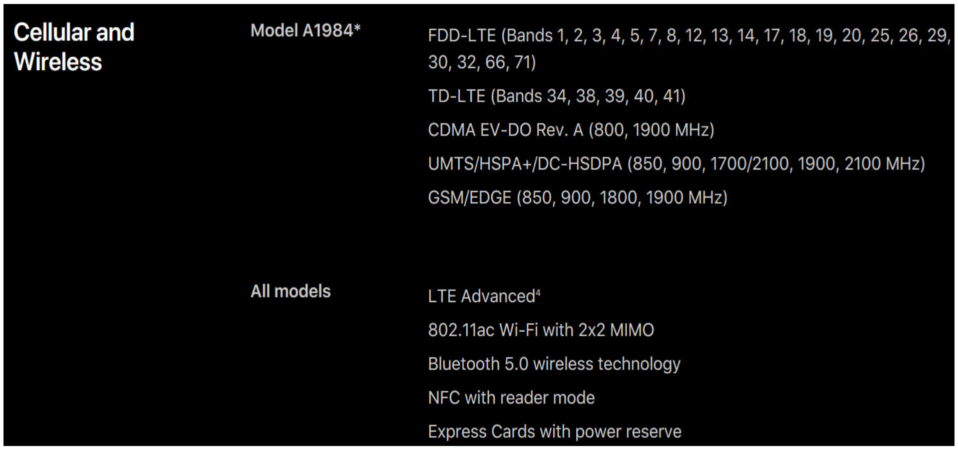
Capacity[1]	64GB 128GB 256GB

https://www.apple.com/iphone-xr/specs/



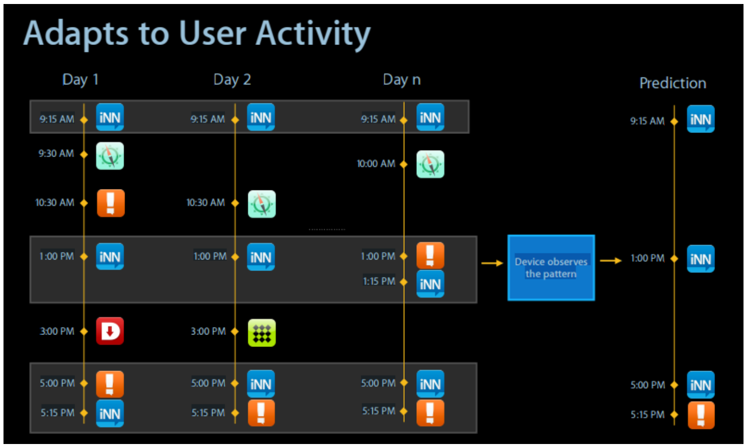
Chip	A12	A12 Bionic chip Next-generation Neural Engine

https://www.apple.com/iphone-xr/specs/



Cellular and Wireless	Model A1984*	FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41) CDMA EV-DO Rev. A (800, 1900 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
	All models	LTE Advanced[4] 802.11ac Wi-Fi with 2x2 MIMO Bluetooth 5.0 wireless technology NFC with reader mode Express Cards with power reserve

https://www.apple.com/iphone-xr/specs/

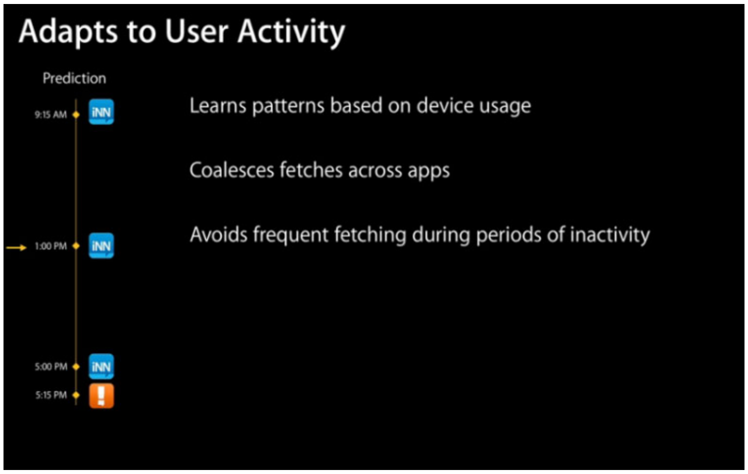


Adapts to User Activity

https://developer.apple.com/videos/wwdc/2013/ ("What's New with Multitasking")

138. Claim 1 of the '056 patent recites "batch data from a first application and a second application for transmission to a respective first application server and a second application server over the wireless network." The '056 accused products satisfy this limitation. For example, the '056 accused devices are configured to batch data from a first application and a second application for transmission to a respective first application server and a second

application server over a wireless network, as shown by the exemplary documentation and developer conference presentations below.



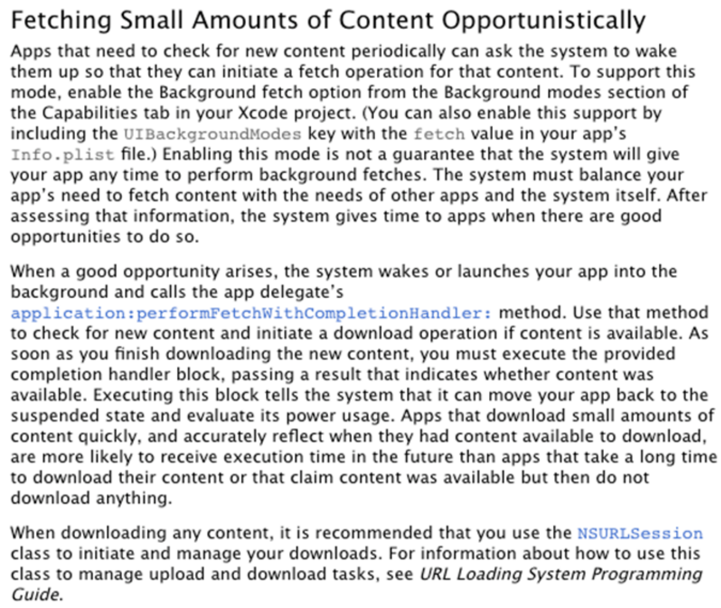
Adapts to User Activity

Learns patterns based on device usage

Coalesces fetches across apps

Avoids frequent fetching during periods of inactivity

https://developer.apple.com/videos/play/wwdc2013/204/



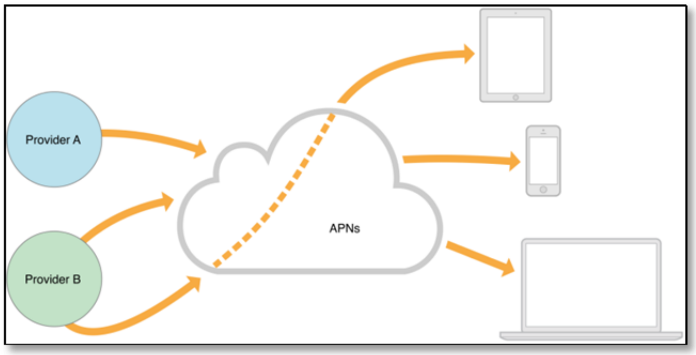
Fetching Small Amounts of Content Opportunistically

Apps that need to check for new content periodically can ask the system to wake them up so that they can initiate a fetch operation for that content. To support this mode, enable the Background fetch option from the Background modes section of the Capabilities tab in your Xcode project. (You can also enable this support by including the `UIBackgroundModes` key with the `fetch` value in your app's `Info.plist` file.) Enabling this mode is not a guarantee that the system will give your app any time to perform background fetches. The system must balance your app's need to fetch content with the needs of other apps and the system itself. After assessing that information, the system gives time to apps when there are good opportunities to do so.

When a good opportunity arises, the system wakes or launches your app into the background and calls the app delegate's `application:performFetchWithCompletionHandler:` method. Use that method to check for new content and initiate a download operation if content is available. As soon as you finish downloading the new content, you must execute the provided completion handler block, passing a result that indicates whether content was available. Executing this block tells the system that it can move your app back to the suspended state and evaluate its power usage. Apps that download small amounts of content quickly, and accurately reflect when they had content available to download, are more likely to receive execution time in the future than apps that take a long time to download their content or that claim content was available but then do not download anything.

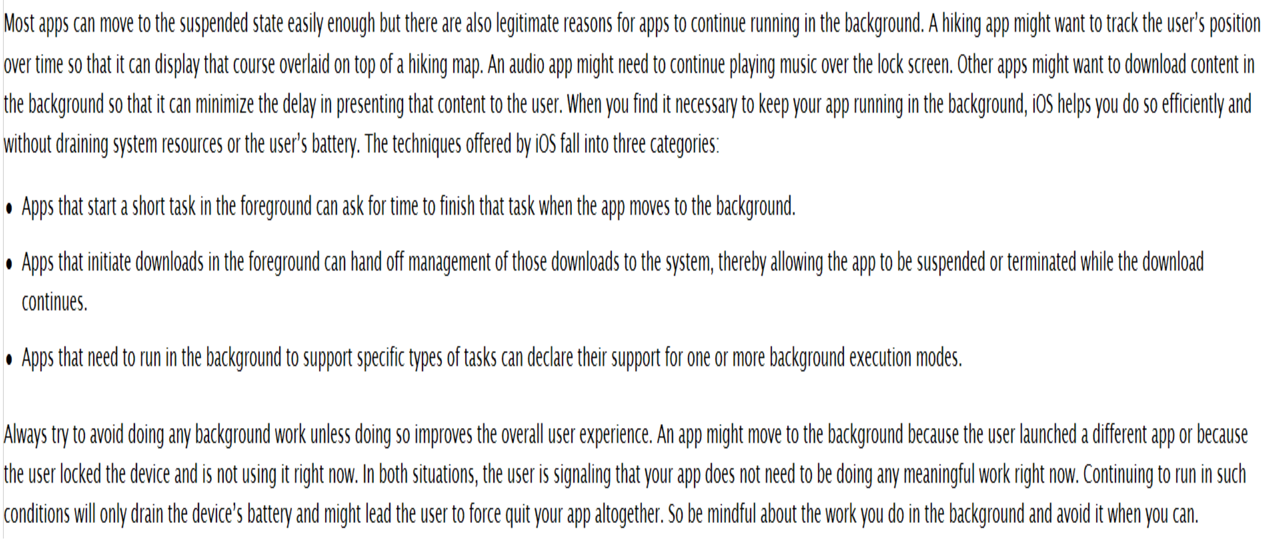
When downloading any content, it is recommended that you use the `NSURLSession` class to initiate and manage your downloads. For information about how to use this class to manage upload and download tasks, see *URL Loading System Programming Guide*.

https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html#//apple_ref/doc/uid/TP40007072-CH4-SW56



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html

139. Claim 1 of the ’056 patent recites “wherein, the batched data from the first application and the second application is batched while a backlight of the mobile device is off in response to inactivity of the mobile device.” The ’056 accused products satisfy this limitation. For example, in iOS devices, batched data from the first application and the second application, is batched while a backlight of the mobile device is off in response to inactivity of the mobile device. For example, background fetch operates when apps are in the background, including when a backlight is off, as shown by the exemplary documentation below.



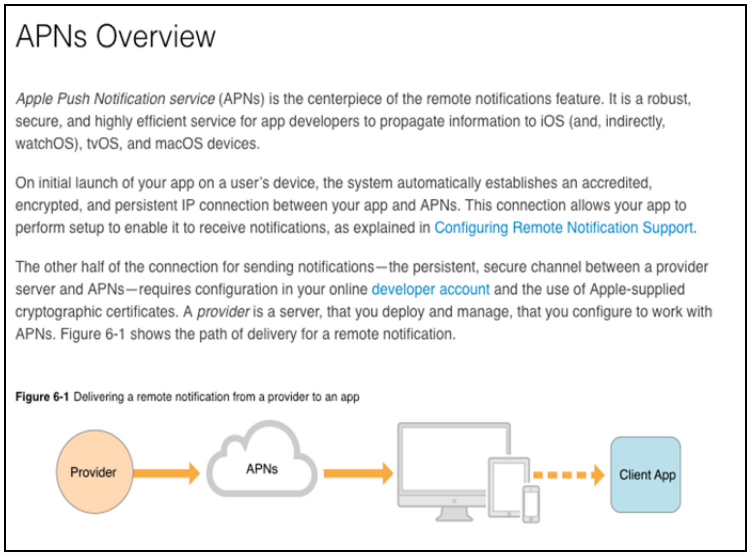
Most apps can move to the suspended state easily enough but there are also legitimate reasons for apps to continue running in the background. A hiking app might want to track the user’s position over time so that it can display that course overlaid on top of a hiking map. An audio app might need to continue playing music over the lock screen. Other apps might want to download content in the background so that it can minimize the delay in presenting that content to the user. When you find it necessary to keep your app running in the background, iOS helps you do so efficiently and without draining system resources or the user’s battery. The techniques offered by iOS fall into three categories:

- Apps that start a short task in the foreground can ask for time to finish that task when the app moves to the background.
- Apps that initiate downloads in the foreground can hand off management of those downloads to the system, thereby allowing the app to be suspended or terminated while the download continues.
- Apps that need to run in the background to support specific types of tasks can declare their support for one or more background execution modes.

Always try to avoid doing any background work unless doing so improves the overall user experience. An app might move to the background because the user launched a different app or because the user locked the device and is not using it right now. In both situations, the user is signaling that your app does not need to be doing any meaningful work right now. Continuing to run in such conditions will only drain the device’s battery and might lead the user to force quit your app altogether. So be mindful about the work you do in the background and avoid it when you can.

https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html#//apple_ref/doc/uid/TP40007072-CH4-SW5

140. Claim 1 of the ’056 patent recites “allow a first message from a remote server distinct from the first application server and the second application server to be received while the batched data from the first application and the second application is batched.” The ’056 accused products satisfy this limitation. For example, the ’056 accused products are configured to allow a first message from a remote server distinct from the first application server and the second application server to be received while the batched data from the first application and the second application is batched, as shown by the exemplary documentation below. As illustrated below, Apple’s Push Notification service (APNs) sends push notifications to iOS devices. It does so using a server (related to for example the cloud below) distinct from an application’s corresponding application server (related to for example the orange circle).



APNs Overview

Apple Push Notification service (APNs) is the centerpiece of the remote notifications feature. It is a robust, secure, and highly efficient service for app developers to propagate information to iOS (and, indirectly, watchOS), tvOS, and macOS devices.

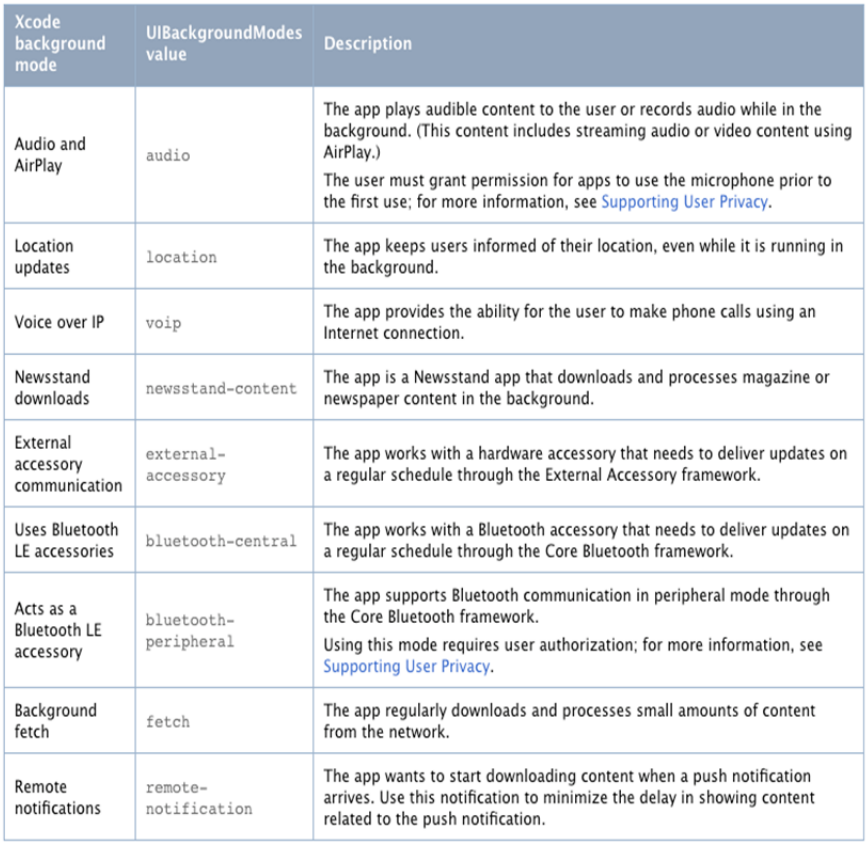
On initial launch of your app on a user’s device, the system automatically establishes an accredited, encrypted, and persistent IP connection between your app and APNs. This connection allows your app to perform setup to enable it to receive notifications, as explained in Configuring Remote Notification Support.

The other half of the connection for sending notifications—the persistent, secure channel between a provider server and APNs—requires configuration in your online developer account and the use of Apple-supplied cryptographic certificates. A *provider* is a server, that you deploy and manage, that you configure to work with APNs. Figure 6-1 shows the path of delivery for a remote notification.

Figure 6-1 Delivering a remote notification from a provider to an app

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

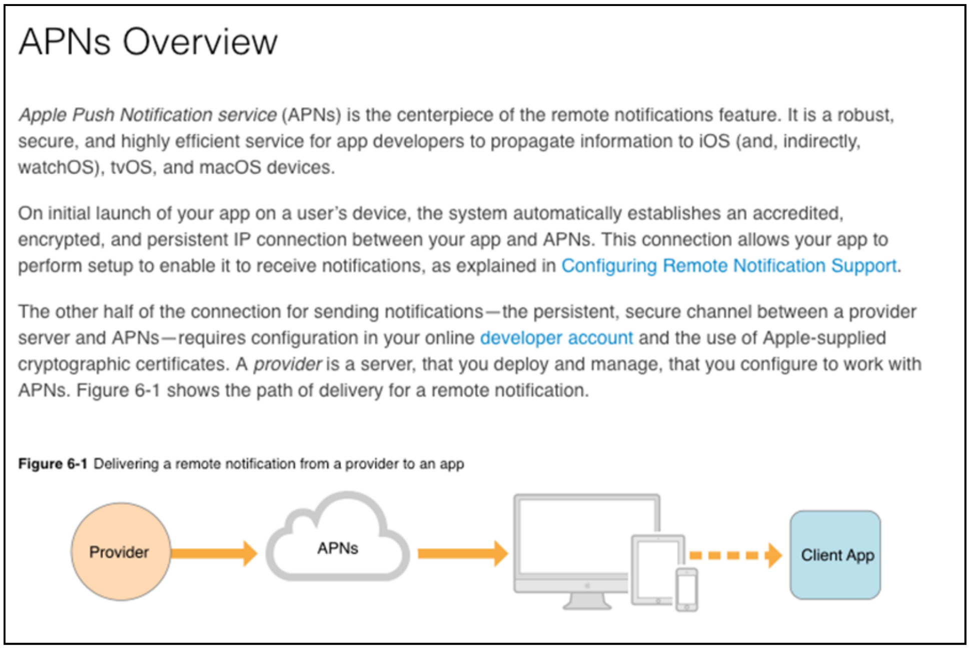
Table 3-1 Background modes for apps



Xcode background mode	UIBackgroundModes value	Description
Audio and AirPlay	`audio`	The app plays audible content to the user or records audio while in the background. (This content includes streaming audio or video content using AirPlay.) The user must grant permission for apps to use the microphone prior to the first use; for more information, see Supporting User Privacy.
Location updates	`location`	The app keeps users informed of their location, even while it is running in the background.
Voice over IP	`voip`	The app provides the ability for the user to make phone calls using an Internet connection.
Newsstand downloads	`newsstand-content`	The app is a Newsstand app that downloads and processes magazine or newspaper content in the background.
External accessory communication	`external-accessory`	The app works with a hardware accessory that needs to deliver updates on a regular schedule through the External Accessory framework.
Uses Bluetooth LE accessories	`bluetooth-central`	The app works with a Bluetooth accessory that needs to deliver updates on a regular schedule through the Core Bluetooth framework.
Acts as a Bluetooth LE accessory	`bluetooth-peripheral`	The app supports Bluetooth communication in peripheral mode through the Core Bluetooth framework. Using this mode requires user authorization; for more information, see Supporting User Privacy.
Background fetch	`fetch`	The app regularly downloads and processes small amounts of content from the network.
Remote notifications	`remote-notification`	The app wants to start downloading content when a push notification arrives. Use this notification to minimize the delay in showing content related to the push notification.

https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html#//apple_ref/doc/uid/TP40007072-CH4-SW56

141. Claim 1 of the ’056 patent recites “wherein the first message from the remote server is directed to the first application and contains data from the first application server and is associated with the mobile device and the first application.” The ’056 accused products satisfy this limitation. For example, in the ’056 accused devices, the first message from the remote server (related to APNs) is directed to the first application and contains data from the first application server and is associated with the mobile device and the first application, as shown by the exemplary documentation below.



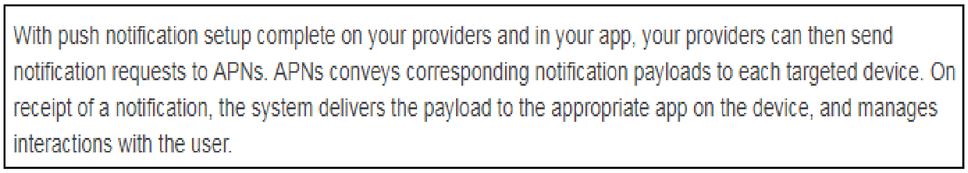
APNs Overview

Apple Push Notification service (APNs) is the centerpiece of the remote notifications feature. It is a robust, secure, and highly efficient service for app developers to propagate information to iOS (and, indirectly, watchOS), tvOS, and macOS devices.

On initial launch of your app on a user's device, the system automatically establishes an accredited, encrypted, and persistent IP connection between your app and APNs. This connection allows your app to perform setup to enable it to receive notifications, as explained in Configuring Remote Notification Support.

The other half of the connection for sending notifications—the persistent, secure channel between a provider server and APNs—requires configuration in your online developer account and the use of Apple-supplied cryptographic certificates. A *provider* is a server, that you deploy and manage, that you configure to work with APNs. Figure 6-1 shows the path of delivery for a remote notification.

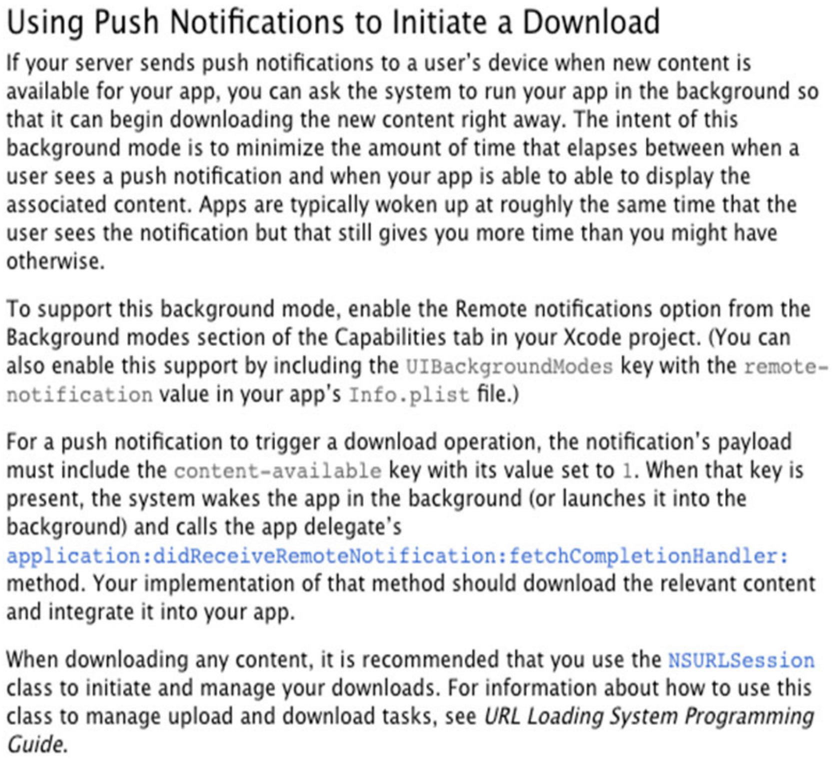
Figure 6-1 Delivering a remote notification from a provider to an app



With push notification setup complete on your providers and in your app, your providers can then send notification requests to APNs. APNs conveys corresponding notification payloads to each targeted device. On receipt of a notification, the system delivers the payload to the appropriate app on the device, and manages interactions with the user.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

142. Claim 1 of the '056 patent recites "transmit a second message associated with the first application to the remote server or the first application server in response to receipt of the first message from the remote server." The '056 accused products satisfy this limitation. For example, the '056 accused products are configured to transmit a second message associated with the first application to the remote server or the first application server in response to receipt of the first message from the remote server, as shown by the exemplary documentation below. For example, the '056 accused products are configured ot send data requests to their corresponding application servers in response to some APNs message types, even when an application is in background or suspended state, as shown by the exemplary documentation below.



Using Push Notifications to Initiate a Download

If your server sends push notifications to a user's device when new content is available for your app, you can ask the system to run your app in the background so that it can begin downloading the new content right away. The intent of this background mode is to minimize the amount of time that elapses between when a user sees a push notification and when your app is able to able to display the associated content. Apps are typically woken up at roughly the same time that the user sees the notification but that still gives you more time than you might have otherwise.

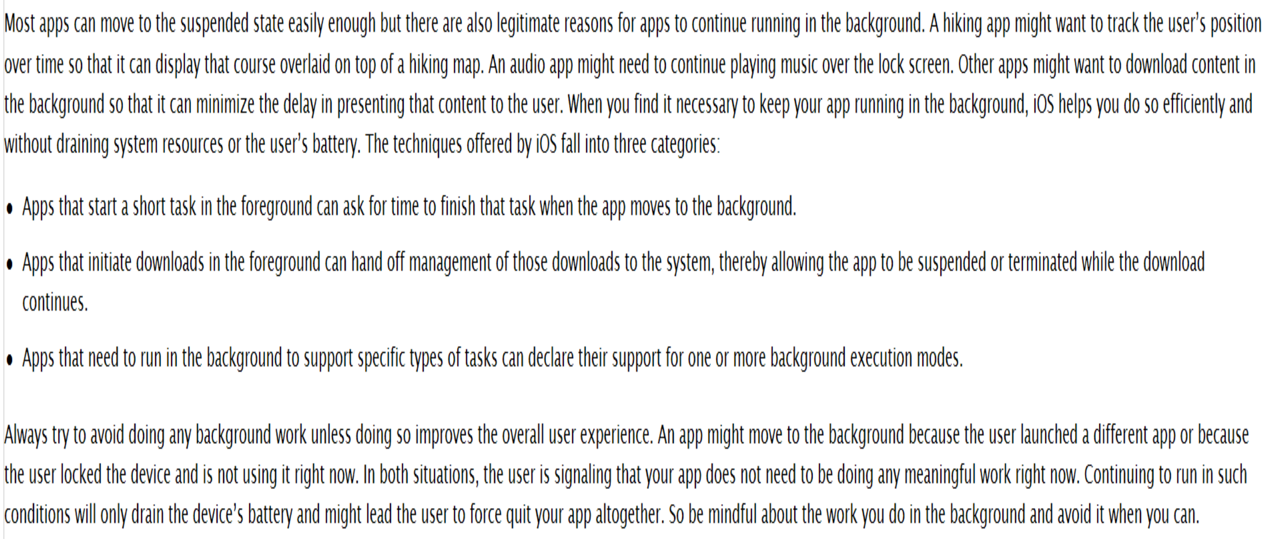
To support this background mode, enable the Remote notifications option from the Background modes section of the Capabilities tab in your Xcode project. (You can also enable this support by including the `UIBackgroundModes` key with the `remote-notification` value in your app's `Info.plist` file.)

For a push notification to trigger a download operation, the notification's payload must include the `content-available` key with its value set to `1`. When that key is present, the system wakes the app in the background (or launches it into the background) and calls the app delegate's `application:didReceiveRemoteNotification:fetchCompletionHandler:` method. Your implementation of that method should download the relevant content and integrate it into your app.

When downloading any content, it is recommended that you use the `NSURLSession` class to initiate and manage your downloads. For information about how to use this class to manage upload and download tasks, see *URL Loading System Programming Guide*.

https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html#//apple_ref/doc/uid/TP40007072-CH4-SW56

143. Claim 1 of the '056 patent recites "transmit the batched data to the respective first application server and the second application server over the wireless network while the backlight of the mobile device remains off." The '056 accused products satisfy this limitation. For example, the '056 accused products are configured to transmit the batched data to the respective first application server and the second application server over the wireless network while the backlight of the mobile device remains off, as shown by the exemplary documentation below. Background fetch requests occur when boths apps are in the background and also when the backlight is off.



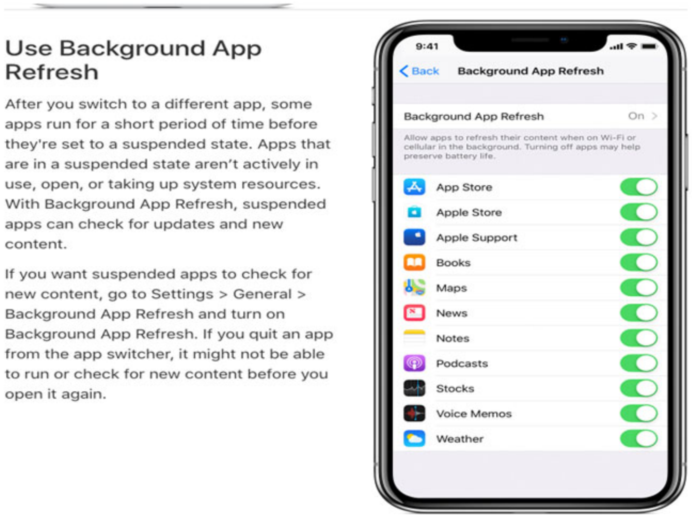
Most apps can move to the suspended state easily enough but there are also legitimate reasons for apps to continue running in the background. A hiking app might want to track the user's position over time so that it can display that course overlaid on top of a hiking map. An audio app might need to continue playing music over the lock screen. Other apps might want to download content in the background so that it can minimize the delay in presenting that content to the user. When you find it necessary to keep your app running in the background, iOS helps you do so efficiently and without draining system resources or the user's battery. The techniques offered by iOS fall into three categories:

- Apps that start a short task in the foreground can ask for time to finish that task when the app moves to the background.
- Apps that initiate downloads in the foreground can hand off management of those downloads to the system, thereby allowing the app to be suspended or terminated while the download continues.
- Apps that need to run in the background to support specific types of tasks can declare their support for one or more background execution modes.

Always try to avoid doing any background work unless doing so improves the overall user experience. An app might move to the background because the user launched a different app or because the user locked the device and is not using it right now. In both situations, the user is signaling that your app does not need to be doing any meaningful work right now. Continuing to run in such conditions will only drain the device's battery and might lead the user to force quit your app altogether. So be mindful about the work you do in the background and avoid it when you can.

https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html#//apple_ref/doc/uid/TP40007072-CH4-SW5

144. Claim 1 of the ’056 patent recites “wherein the batching of data for the first application and the second application can be enabled or disabled by a user of the mobile device on an application-by-application basis.” The ’056 accused products satisfy this limitation. For example, in the ’056 accused products, the batching of data for the first application and the second application can be enabled or disabled by a user of the mobile device on an application-by-application basis, as shown by the exemplary documentation below.



Use Background App Refresh

After you switch to a different app, some apps run for a short period of time before they're set to a suspended state. Apps that are in a suspended state aren't actively in use, open, or taking up system resources. With Background App Refresh, suspended apps can check for updates and new content.

If you want suspended apps to check for new content, go to Settings > General > Background App Refresh and turn on Background App Refresh. If you quit an app from the app switcher, it might not be able to run or check for new content before you open it again.

Background App Refresh On >

Allow apps to refresh their content when on Wi-Fi or cellular in the background. Turning off apps may help preserve battery life.

- App Store ☑
- Apple Store ☑
- Apple Support ☑
- Books ☑
- Maps ☑
- News ☑
- Notes ☑
- Podcasts ☑
- Stocks ☑
- Voice Memos ☑
- Weather ☑

https://support.apple.com/en-us/HT202070

145. Thus, as illustrated above, the ’056 accused products directly infringe one or more claims of the ’056 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the ’056 patent.

146. Apple indirectly infringes the ’056 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple’s customers and end-users, in this district and elsewhere in the United States. For example, Apple’s customers and end-users directly infringe through their use of the inventions claimed in the ’056 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’056 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the ’056 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products’ capability in an infringing manner, including through the use of Apple’s background app data fetching and background app refresh. *See, e.g.*, https://support.apple.com/en-us/HT202070; https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html#//apple_ref/doc/uid/TP40007072-CH4-SW56. As a result of Apple’s inducement, Apple’s customers and end-users use the

Apple products in the way Apple intends and directly infringe the ’056 patent. Apple performs these affirmative acts with knowledge of the ’056 patent and with the intent, or willful blindness, that the induced acts directly infringe the ’056 patent. Apple has had knowledge and notice of the ’056 patent at least as of the filing of this complaint.

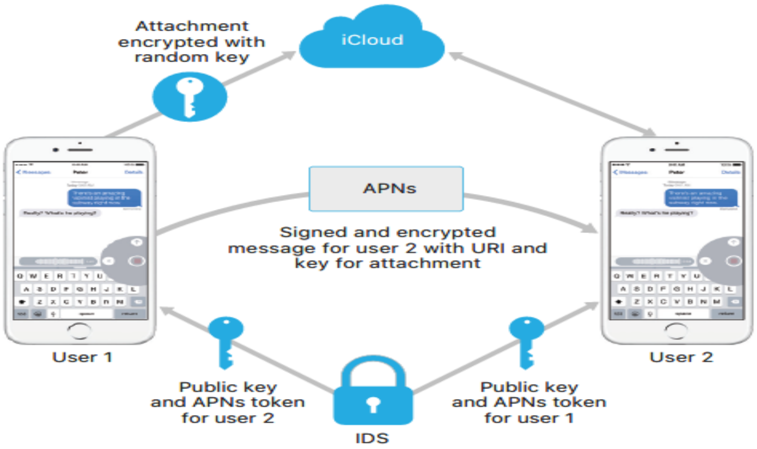
147. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

COUNT VI: PATENT INFRINGEMENT OF THE ’968 PATENT

148. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

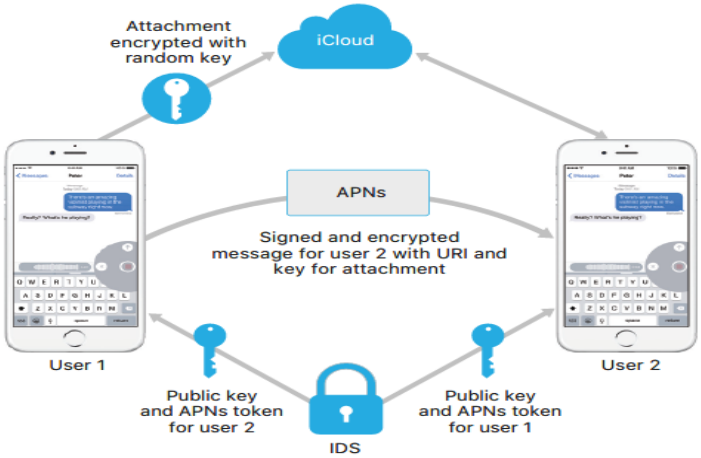
149. Apple infringes (literally and/or under the Doctrine of Equivalents) the ’968 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the ’968 patent including at least the ’968 accused products noted above. In addition, Apple derives revenue from the activities relating to the ’968 accused products.

150. For example and as shown below, the ’968 accused products infringe at least claim 1 of the ’968 patent. For example, and to the extent the preamble is limiting, the ’968 accused products satisfy a “server that manages transactions between first and second devices, the server comprising: a communication interface; a processor communicatively coupled to the communication interface; and a memory communicatively coupled to the processor, the memory containing instructions executable by the processor whereby the server is operable to:” As shown below by the exemplary evidence, the ’968 accused products that manage transactions between first and second devices (e.g., iPhones, iPads, etc.) comprising a communication interface, a memory, and a processor.


Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
IDS
Public key and APNs token for user 1

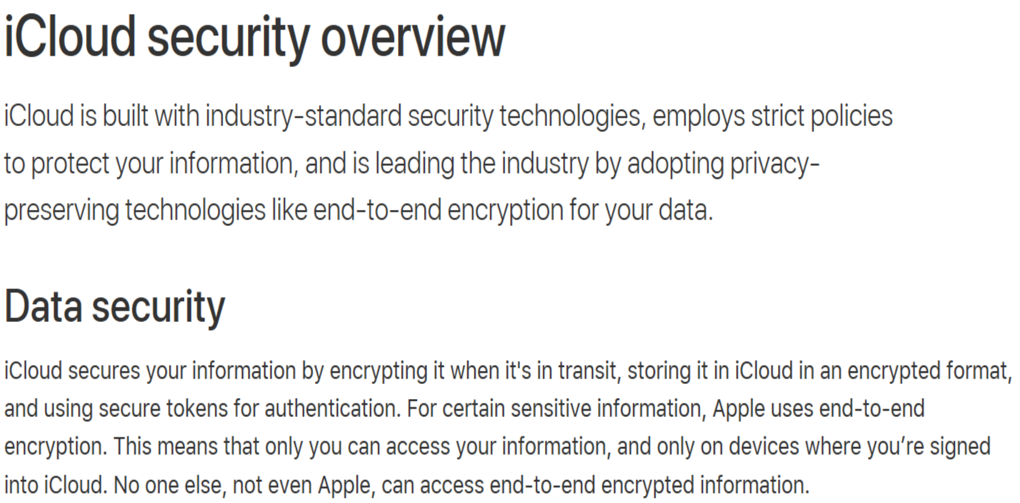
https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

151. Claim 1 of the ’968 patent recites “receive a first connection from a first device.” The ’968 accused products satisfy this limitation. For example, the Apple server receives a connections from a first device (e.g., iPhone, iPad, Mac, etc.), as shown in the exemplary documentation below.


Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
IDS
Public key and APNs token for user 1

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

152. Claim 1 of the ’968 patent recites “authenticate the first device over a first connection.” The ’968 accused products satisfy this limitation. For example, the Apple server authenticates the first device over a first connection.



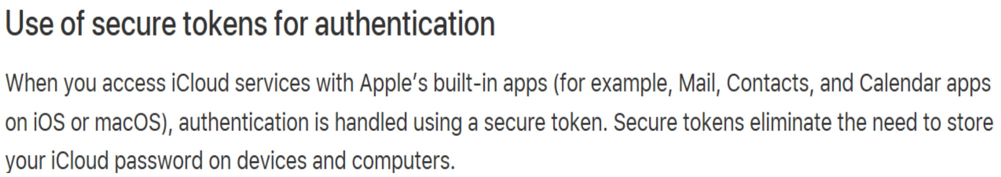
iCloud security overview

iCloud is built with industry-standard security technologies, employs strict policies to protect your information, and is leading the industry by adopting privacy-preserving technologies like end-to-end encryption for your data.

Data security

iCloud secures your information by encrypting it when it's in transit, storing it in iCloud in an encrypted format, and using secure tokens for authentication. For certain sensitive information, Apple uses end-to-end encryption. This means that only you can access your information, and only on devices where you're signed into iCloud. No one else, not even Apple, can access end-to-end encrypted information.

****

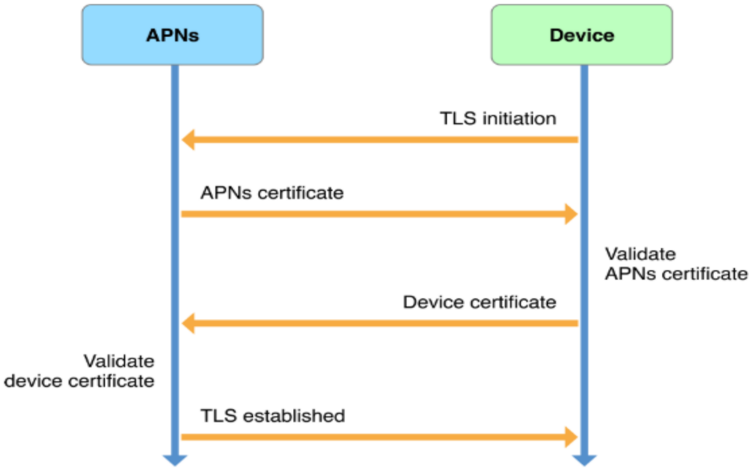


Use of secure tokens for authentication

When you access iCloud services with Apple's built-in apps (for example, Mail, Contacts, and Calendar apps on iOS or macOS), authentication is handled using a secure token. Secure tokens eliminate the need to store your iCloud password on devices and computers.

https://support.apple.com/en-us/HT202303

Figure 6-5 Establishing connection trust between a device and APNs

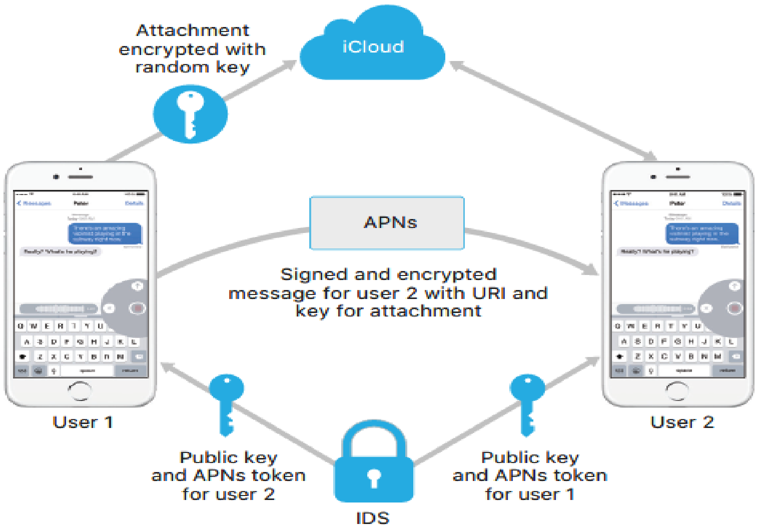


https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

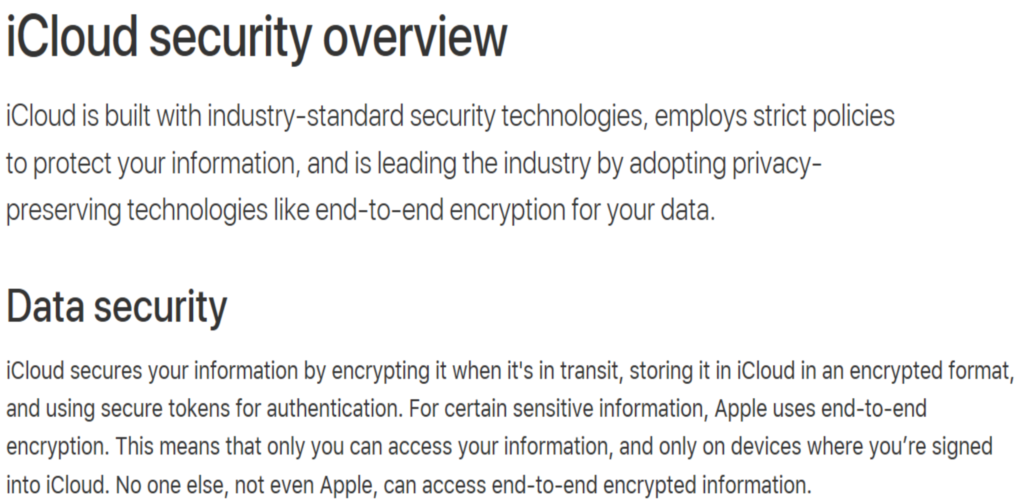
```
default 20:40:27.554127 -0400  apsd   2019-03-16 20:40:27 -0400 apsd[85]: <APSCourier: 0x11bd0d530>: Sending filter message for: token type systemToken, token: <6dc220e2 a40b027a d01c77ce 7df3f958 fdbb9ec5 bf218587 fc444b14 e2e9c4b1>, enabled hashes {
    <307ce9d8 26877944 a0cda5fb 151965fe ad398f9f> = "com.spotify.client";
    <bb5ddb5d 1674248f 6a18c0b9 ec4fec0b a9477d60> = "com.apple.maps.icloud";
    <63a82ad5 e27b04a3 5cefda08 e3773f25 adf3b8d6> = "com.me.keyvalueservice";
    <0d77daf0 96ceb132 655f3a29 12118ce8 609026a3> = "us.parkmobile.ParkMobile";
    <fa3049fc 04e3afe0 752e2938 8f086616 16faac92> = "com.google.hangouts.voip";
    <a703ff59 e0d4dfe6 1fba112c 4f744c38 5fae93ce> = "com.facebook.Messenger.voip";
    <6e2f732d d6872e1f 58f193dd 18638d18 0b78c5a0> = "com.ubercab.UberEats";
    <a1d17919 7ab41a91 59e473b6 0c8ddcf9 4ed8e840> = "jp.naver.line";
    <e82aba2c 4933a627 094d2ed6 4f60f3f6 2e288afa> = "com.amtrak.rider";
    <e7e89581 8ca86c28 863dce40 2c5092b9 8ac35b20> = "com.skype.skype";
    <f9b12762 c0dcf6f0 da651a65<…>
default 20:40:27.554586 -0400 apsd   2019-03-16 20:40:27 -0400 apsd[85]: Stopped indexing
hashes at index 38 {token: <6dc220e2 a40b027a d01c77ce 7df3f958 fdbb9ec5 bf218587
fc444b14 e2e9c4b1>, enabledHashes.count: 110, ignoredHashes.count: 1,
opportunisticHashes.count: 95, pausedHashes.count: 0}
```

iPhone log captured March 16, 2019

153. Claim 1 of the ’968 patent recites “receive a first transaction from the first device in response to user input at the first device, wherein the first connection is maintained independently of the first transaction.” The ’968 accused products satisfy this limitation. The Apple server receives a first transaction from a first device (e.g., iPhone, iPad, etc) and the connection is maintained independently of the first transaction, as shown by the exemplary documentation below.



https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf



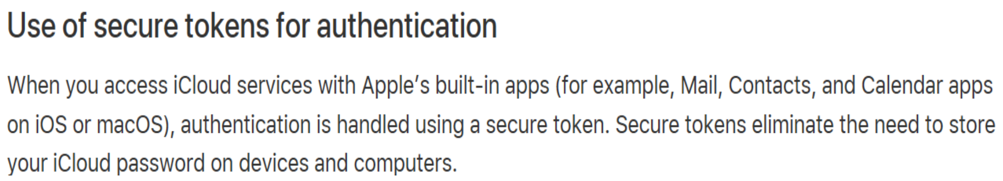
iCloud security overview

iCloud is built with industry-standard security technologies, employs strict policies to protect your information, and is leading the industry by adopting privacy-preserving technologies like end-to-end encryption for your data.

Data security

iCloud secures your information by encrypting it when it's in transit, storing it in iCloud in an encrypted format, and using secure tokens for authentication. For certain sensitive information, Apple uses end-to-end encryption. This means that only you can access your information, and only on devices where you're signed into iCloud. No one else, not even Apple, can access end-to-end encrypted information.

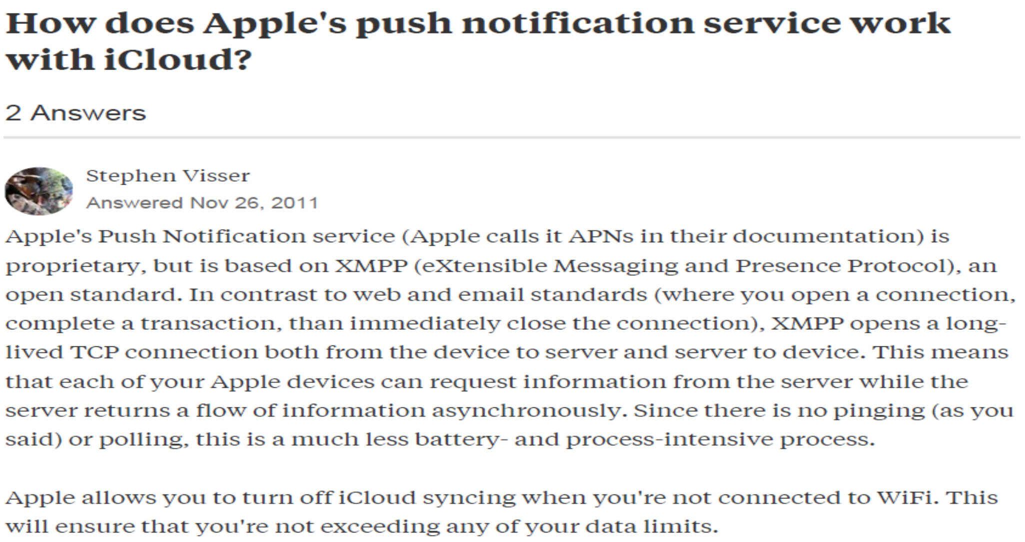
****



Use of secure tokens for authentication

When you access iCloud services with Apple's built-in apps (for example, Mail, Contacts, and Calendar apps on iOS or macOS), authentication is handled using a secure token. Secure tokens eliminate the need to store your iCloud password on devices and computers.

https://support.apple.com/en-us/HT202303



How does Apple's push notification service work with iCloud?

2 Answers

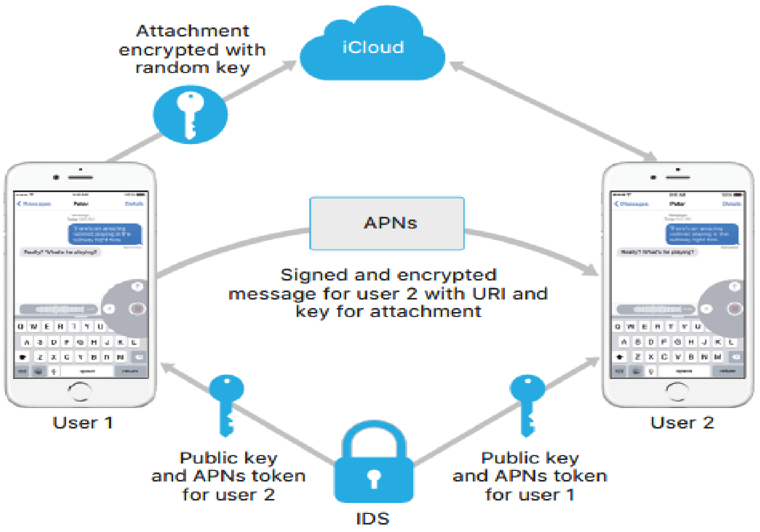
Stephen Visser
Answered Nov 26, 2011

Apple's Push Notification service (Apple calls it APNs in their documentation) is proprietary, but is based on XMPP (eXtensible Messaging and Presence Protocol), an open standard. In contrast to web and email standards (where you open a connection, complete a transaction, than immediately close the connection), XMPP opens a long-lived TCP connection both from the device to server and server to device. This means that each of your Apple devices can request information from the server while the server returns a flow of information asynchronously. Since there is no pinging (as you said) or polling, this is a much less battery- and process-intensive process.

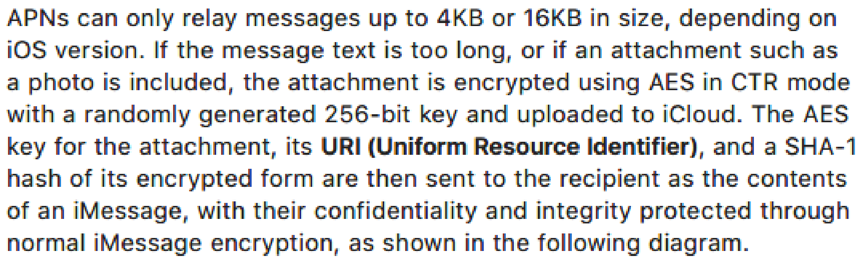
Apple allows you to turn off iCloud syncing when you're not connected to WiFi. This will ensure that you're not exceeding any of your data limits.

https://www.quora.com/How-does-Apples-push-notification-service-work-with-iCloud

154. Claim 1 of the '968 patent recites "generate a trigger for a second device based on the first transaction from the first device, wherein the trigger is pushed to the second device." The '968 accused products satisfy this limitation. For example, the Apple server generates a trigger for a second device, and the trigger is pushed to the second device, as shown in the exemplary documentation below.

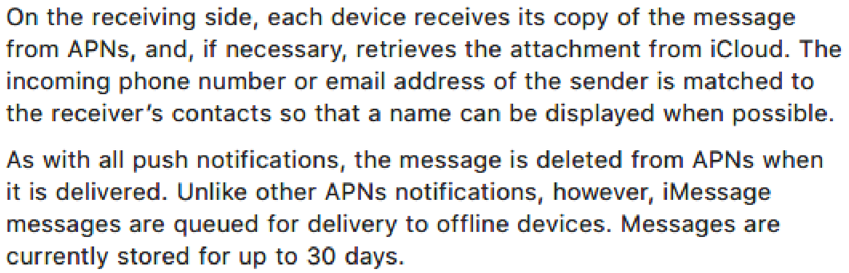

Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
IDS
Public key and APNs token for user 1

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf



APNs can only relay messages up to 4KB or 16KB in size, depending on iOS version. If the message text is too long, or if an attachment such as a photo is included, the attachment is encrypted using AES in CTR mode with a randomly generated 256-bit key and uploaded to iCloud. The AES key for the attachment, its **URI (Uniform Resource Identifier)**, and a SHA-1 hash of its encrypted form are then sent to the recipient as the contents of an iMessage, with their confidentiality and integrity protected through normal iMessage encryption, as shown in the following diagram.

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf



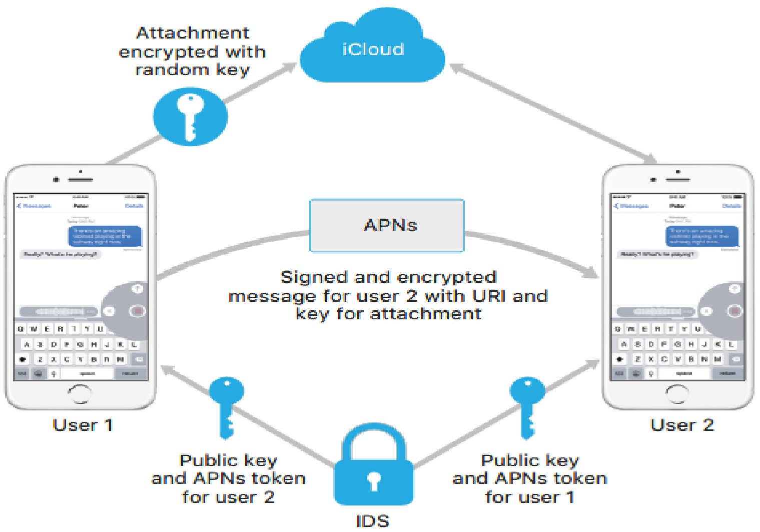
On the receiving side, each device receives its copy of the message from APNs, and, if necessary, retrieves the attachment from iCloud. The incoming phone number or email address of the sender is matched to the receiver's contacts so that a name can be displayed when possible.

As with all push notifications, the message is deleted from APNs when it is delivered. Unlike other APNs notifications, however, iMessage messages are queued for delivery to offline devices. Messages are currently stored for up to 30 days.

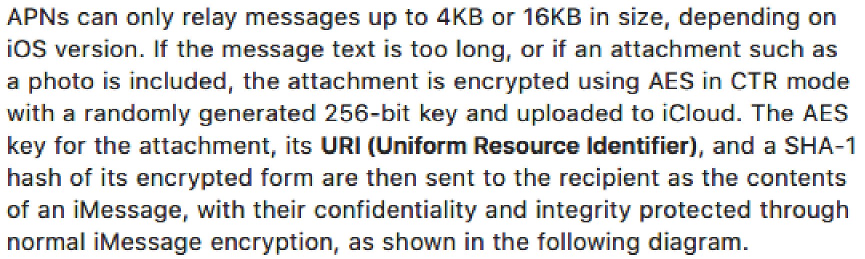
https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

155. Claim 1 of the '968 patent recites "after the generation of the trigger for the second device, receive a second connection from the second device while the first connection is maintained." The '968 accused products satisfy this limitation. For example, after the generation of the trigger for the second device, the Apple server receives a second connection from the second iPhone while the first connection is still maintained, as shown by the exemplary

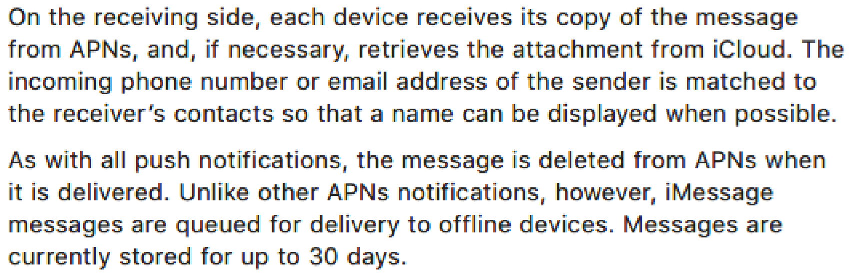
documentation below. The second connection is used for receiving new data, as shown in the exemplary documentation below.


Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf


APNs can only relay messages up to 4KB or 16KB in size, depending on iOS version. If the message text is too long, or if an attachment such as a photo is included, the attachment is encrypted using AES in CTR mode with a randomly generated 256-bit key and uploaded to iCloud. The AES key for the attachment, its **URI (Uniform Resource Identifier)**, and a SHA-1 hash of its encrypted form are then sent to the recipient as the contents of an iMessage, with their confidentiality and integrity protected through normal iMessage encryption, as shown in the following diagram.

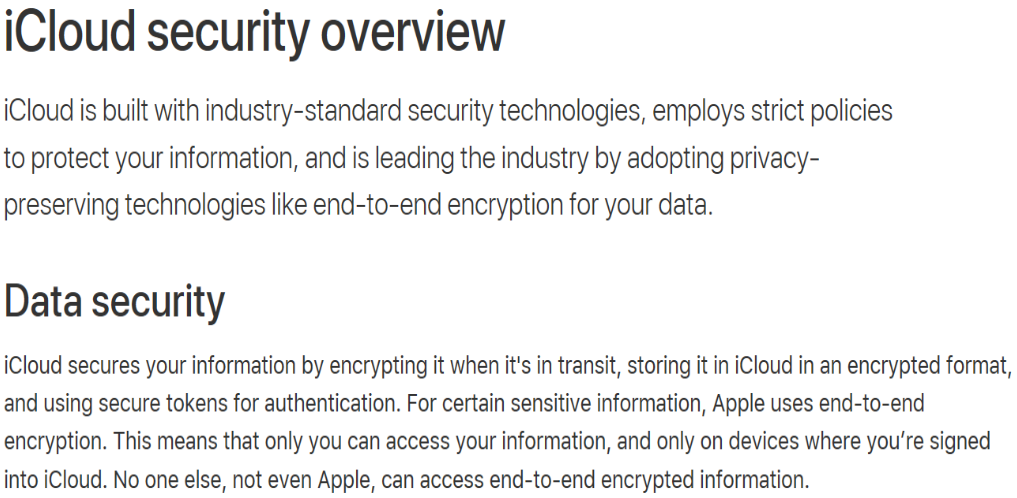
https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf


On the receiving side, each device receives its copy of the message from APNs, and, if necessary, retrieves the attachment from iCloud. The incoming phone number or email address of the sender is matched to the receiver's contacts so that a name can be displayed when possible.

As with all push notifications, the message is deleted from APNs when it is delivered. Unlike other APNs notifications, however, iMessage messages are queued for delivery to offline devices. Messages are currently stored for up to 30 days.

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

156. Claim 1 of the '968 patent recites "authenticate the second device over the second connection." The '968 accused products satisfy this limitation. For example, the Apple server authenticates the second device.



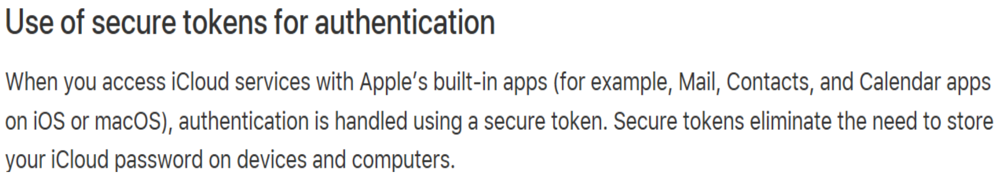
iCloud security overview

iCloud is built with industry-standard security technologies, employs strict policies to protect your information, and is leading the industry by adopting privacy-preserving technologies like end-to-end encryption for your data.

Data security

iCloud secures your information by encrypting it when it's in transit, storing it in iCloud in an encrypted format, and using secure tokens for authentication. For certain sensitive information, Apple uses end-to-end encryption. This means that only you can access your information, and only on devices where you're signed into iCloud. No one else, not even Apple, can access end-to-end encrypted information.

****

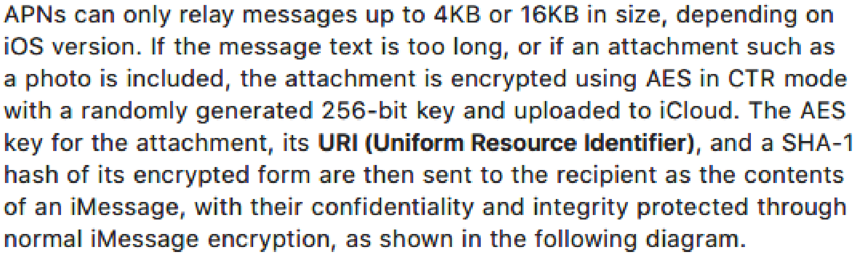


Use of secure tokens for authentication

When you access iCloud services with Apple's built-in apps (for example, Mail, Contacts, and Calendar apps on iOS or macOS), authentication is handled using a secure token. Secure tokens eliminate the need to store your iCloud password on devices and computers.

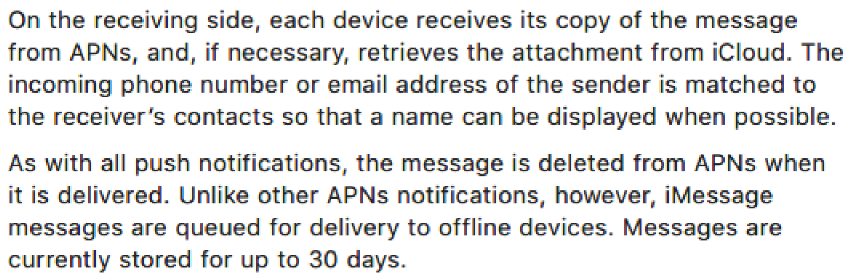
https://support.apple.com/en-us/HT202303

157. Claim 1 of the '968 patent recites "wherein the trigger notifies the second device of new data from the first transaction to be received by the second device from the server for display to a user." The '968 accused products satisfy this limitation. For example, the trigger notifies the second device of new data from the first transaction to be received by the second device from the server for display to a user.



APNs can only relay messages up to 4KB or 16KB in size, depending on iOS version. If the message text is too long, or if an attachment such as a photo is included, the attachment is encrypted using AES in CTR mode with a randomly generated 256-bit key and uploaded to iCloud. The AES key for the attachment, its **URI (Uniform Resource Identifier)**, and a SHA-1 hash of its encrypted form are then sent to the recipient as the contents of an iMessage, with their confidentiality and integrity protected through normal iMessage encryption, as shown in the following diagram.

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

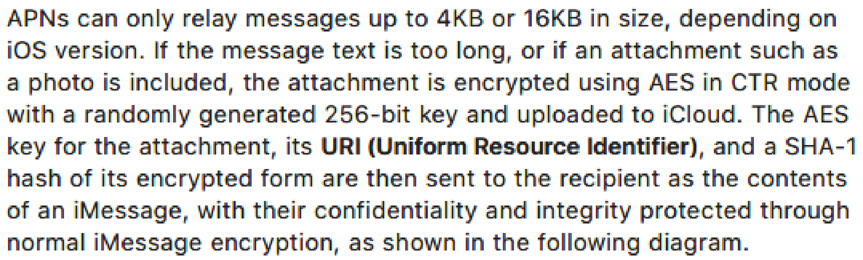


On the receiving side, each device receives its copy of the message from APNs, and, if necessary, retrieves the attachment from iCloud. The incoming phone number or email address of the sender is matched to the receiver's contacts so that a name can be displayed when possible.

As with all push notifications, the message is deleted from APNs when it is delivered. Unlike other APNs notifications, however, iMessage messages are queued for delivery to offline devices. Messages are currently stored for up to 30 days.

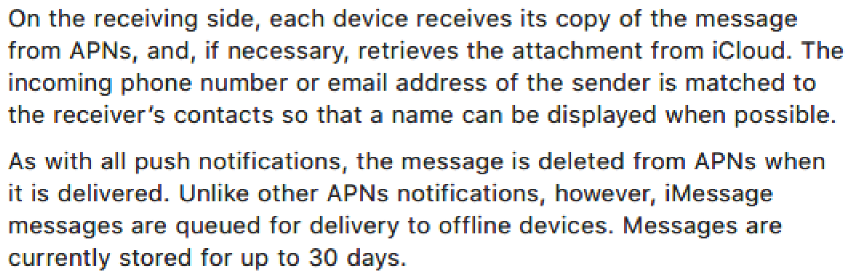
https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

158. Claim 1 of the '968 patent recites "send a second transaction to the second device using the second connection, wherein the second transaction contains the new data." The '968 accused products satisfy this limitation. For example, the Apple server sends a second transaction containing new data to the second device using the second connection.



APNs can only relay messages up to 4KB or 16KB in size, depending on iOS version. If the message text is too long, or if an attachment such as a photo is included, the attachment is encrypted using AES in CTR mode with a randomly generated 256-bit key and uploaded to iCloud. The AES key for the attachment, its **URI (Uniform Resource Identifier)**, and a SHA-1 hash of its encrypted form are then sent to the recipient as the contents of an iMessage, with their confidentiality and integrity protected through normal iMessage encryption, as shown in the following diagram.

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

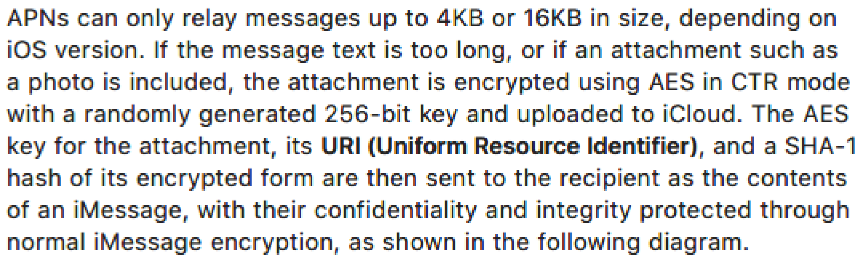


On the receiving side, each device receives its copy of the message from APNs, and, if necessary, retrieves the attachment from iCloud. The incoming phone number or email address of the sender is matched to the receiver's contacts so that a name can be displayed when possible.

As with all push notifications, the message is deleted from APNs when it is delivered. Unlike other APNs notifications, however, iMessage messages are queued for delivery to offline devices. Messages are currently stored for up to 30 days.

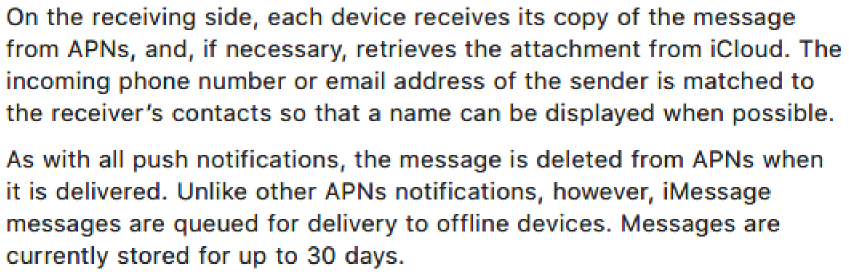
https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

159. Claim 1 of the '968 patent recites "wherein the trigger is pushed over a connection different from the second connection." The '968 accused products satisfy this limitation. For example, the trigger goes over a different connection from the connection the new data goes over.



APNs can only relay messages up to 4KB or 16KB in size, depending on iOS version. If the message text is too long, or if an attachment such as a photo is included, the attachment is encrypted using AES in CTR mode with a randomly generated 256-bit key and uploaded to iCloud. The AES key for the attachment, its **URI (Uniform Resource Identifier)**, and a SHA-1 hash of its encrypted form are then sent to the recipient as the contents of an iMessage, with their confidentiality and integrity protected through normal iMessage encryption, as shown in the following diagram.

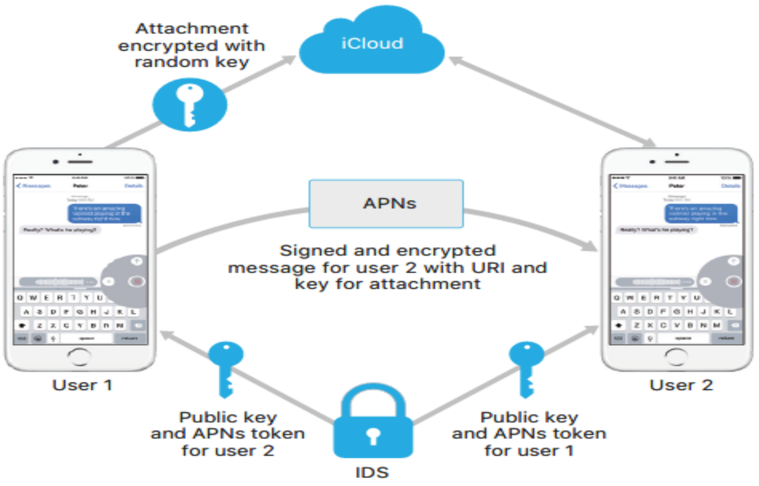
https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf



On the receiving side, each device receives its copy of the message from APNs, and, if necessary, retrieves the attachment from iCloud. The incoming phone number or email address of the sender is matched to the receiver's contacts so that a name can be displayed when possible.

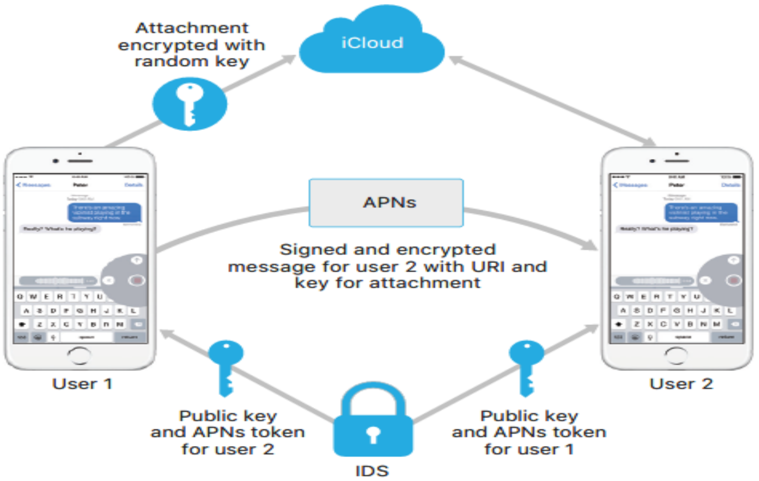
As with all push notifications, the message is deleted from APNs when it is delivered. Unlike other APNs notifications, however, iMessage messages are queued for delivery to offline devices. Messages are currently stored for up to 30 days.

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

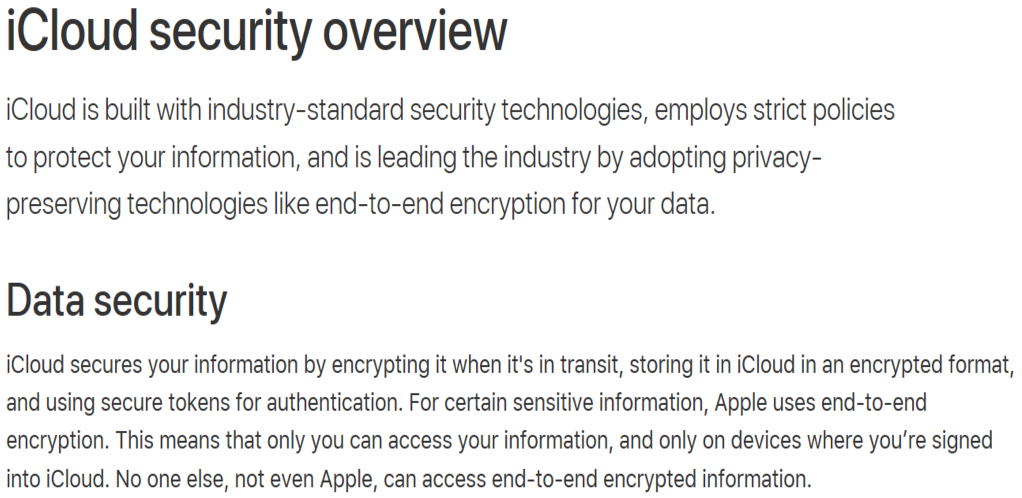

Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
IDS
Public key and APNs token for user 1

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

160. Claim 1 of the ’968 patent recites “wherein a third transaction is received by the server in response to user input at the second device.” The ’968 accused products satisfy this limitation. For example, the second device can send a third transaction to the Apple server, as shown by the exemplary documentation below.


Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf



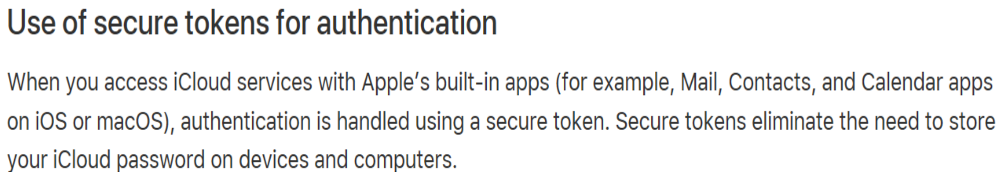
iCloud security overview

iCloud is built with industry-standard security technologies, employs strict policies to protect your information, and is leading the industry by adopting privacy-preserving technologies like end-to-end encryption for your data.

Data security

iCloud secures your information by encrypting it when it's in transit, storing it in iCloud in an encrypted format, and using secure tokens for authentication. For certain sensitive information, Apple uses end-to-end encryption. This means that only you can access your information, and only on devices where you're signed into iCloud. No one else, not even Apple, can access end-to-end encrypted information.

****



Use of secure tokens for authentication

When you access iCloud services with Apple's built-in apps (for example, Mail, Contacts, and Calendar apps on iOS or macOS), authentication is handled using a secure token. Secure tokens eliminate the need to store your iCloud password on devices and computers.

https://support.apple.com/en-us/HT202303

161. Thus, as illustrated above, the ’968 accused products directly infringe one or more claims of the ’968 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the ’968 patent.

162. Apple indirectly infringes the ’968 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple’s customers and end-users, in this district and elsewhere in the United States. For example, Apple’s customers and end-users directly infringe through their use of the inventions claimed in the ’968 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’968 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the ’968 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products’ capability in an infringing manner, including through the use of iCloud and iMessage. *See, e.g.*, https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf. As a result of Apple’s inducement, Apple’s customers and end-users use the Apple products in the way Apple intends and directly infringe the ’968 patent. Apple performs these affirmative acts with knowledge of the ’968 patent and with the intent, or willful blindness, that the induced acts directly infringe the ’968 patent. Apple has had knowledge and notice of the ’968 patent at least as of the filing of this complaint.

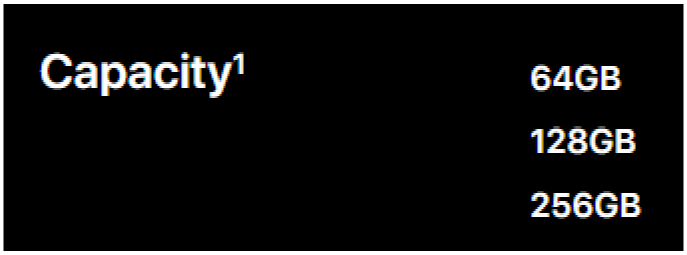
163. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

COUNT VII: PATENT INFRINGEMENT OF THE ’557 PATENT

164. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

165. Apple infringes (literally and/or under the Doctrine of Equivalents) the ’557 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the ’557 patent including at least the ’557 accused products noted above. In addition, Apple derives revenue from the activities relating to the ’557 accused products.

166. For example and as shown below, the ’557 accused products infringe at least claim 14 of the ’557 patent. For example, and to the extent the preamble is limiting, the ’557 accused products satisfy a “mobile device comprising: a memory and a processor; a network interface operable to: connecting to a WIFI network and a cellular network:” As shown below by the exemplary evidence, the ’557 accused products are mobile devices comprising a network interface to connect to a WIFI network and a cellular network, a memory, and a processor.



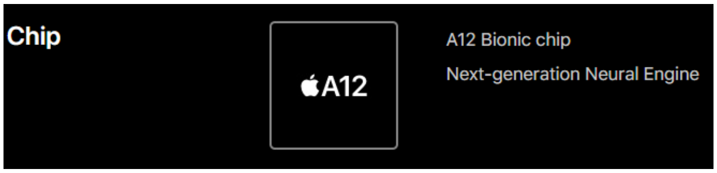
Capacity[1]

64GB

128GB

256GB

https://www.apple.com/iphone-xr/specs/



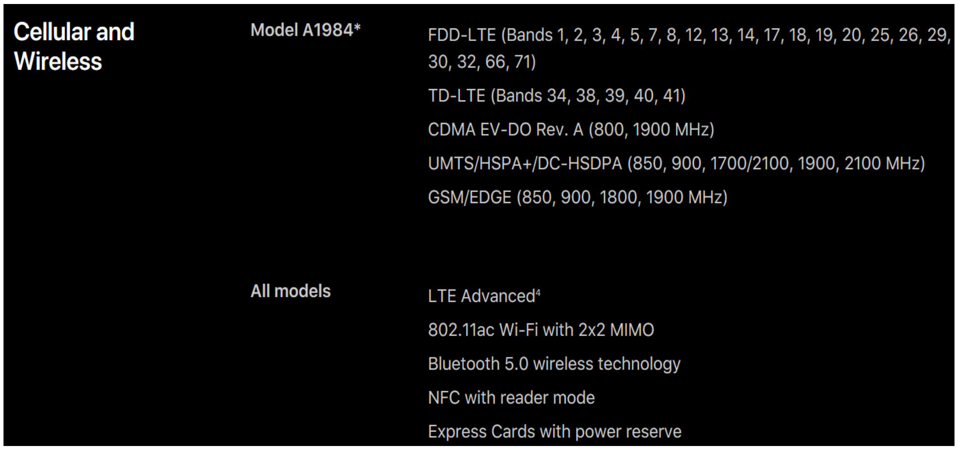
Chip

A12

A12 Bionic chip

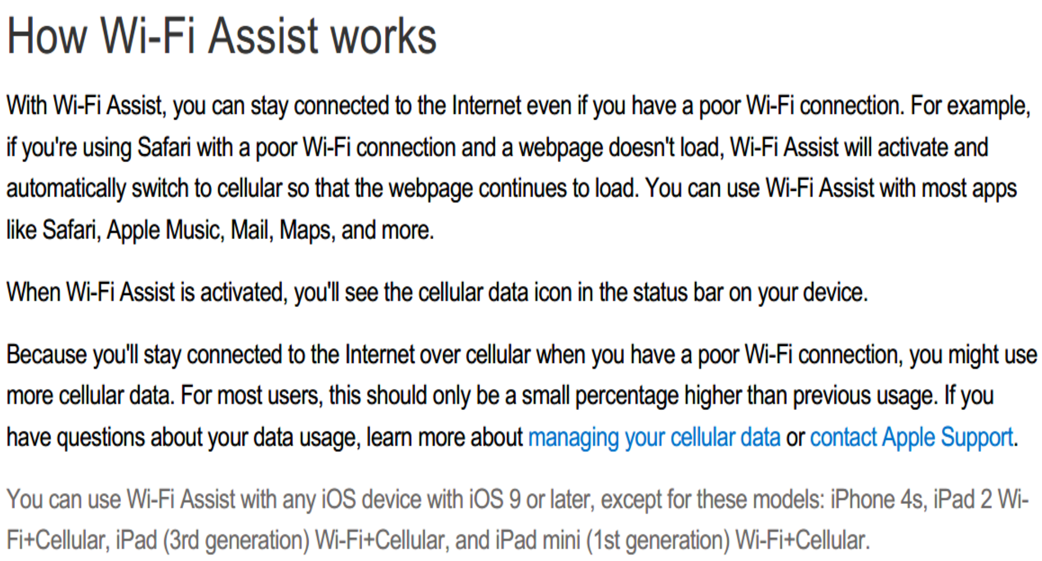
Next-generation Neural Engine

https://www.apple.com/iphone-xr/specs/



Cellular and Wireless	Model A1984*	FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41) CDMA EV-DO Rev. A (800, 1900 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
	All models	LTE Advanced[4] 802.11ac Wi-Fi with 2x2 MIMO Bluetooth 5.0 wireless technology NFC with reader mode Express Cards with power reserve

https://www.apple.com/iphone-xr/specs/



How Wi-Fi Assist works

With Wi-Fi Assist, you can stay connected to the Internet even if you have a poor Wi-Fi connection. For example, if you're using Safari with a poor Wi-Fi connection and a webpage doesn't load, Wi-Fi Assist will activate and automatically switch to cellular so that the webpage continues to load. You can use Wi-Fi Assist with most apps like Safari, Apple Music, Mail, Maps, and more.

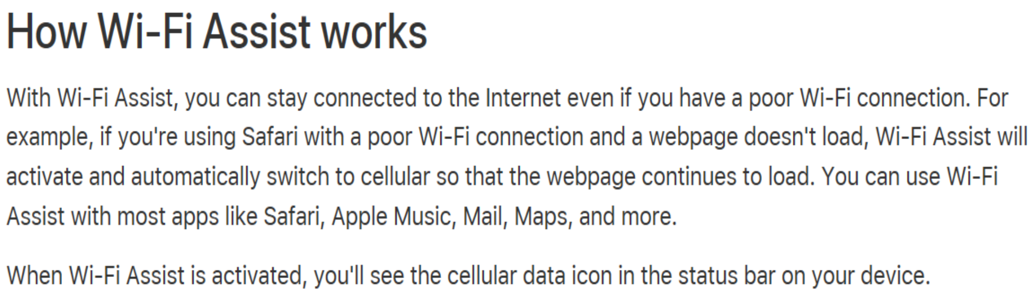
When Wi-Fi Assist is activated, you'll see the cellular data icon in the status bar on your device.

Because you'll stay connected to the Internet over cellular when you have a poor Wi-Fi connection, you might use more cellular data. For most users, this should only be a small percentage higher than previous usage. If you have questions about your data usage, learn more about managing your cellular data or contact Apple Support.

You can use Wi-Fi Assist with any iOS device with iOS 9 or later, except for these models: iPhone 4s, iPad 2 Wi-Fi+Cellular, iPad (3rd generation) Wi-Fi+Cellular, and iPad mini (1st generation) Wi-Fi+Cellular.

https://support.apple.com/en-us/HT205296

167. Claim 14 of the ’557 patent recites “displaying an indication of availability of the WIFI network and the cellular network.” The ’557 accused products satisfy this limitation. For example, the ’557 accused products display indications of the availability of Wi-Fi and the cellular network, as shown by the exemplary documentation below.

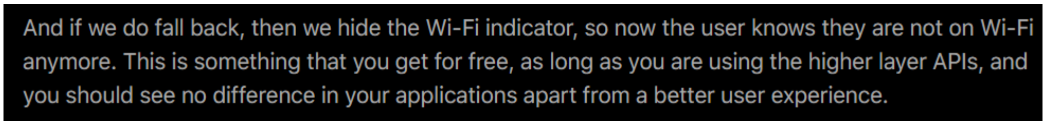


How Wi-Fi Assist works

With Wi-Fi Assist, you can stay connected to the Internet even if you have a poor Wi-Fi connection. For example, if you're using Safari with a poor Wi-Fi connection and a webpage doesn't load, Wi-Fi Assist will activate and automatically switch to cellular so that the webpage continues to load. You can use Wi-Fi Assist with most apps like Safari, Apple Music, Mail, Maps, and more.

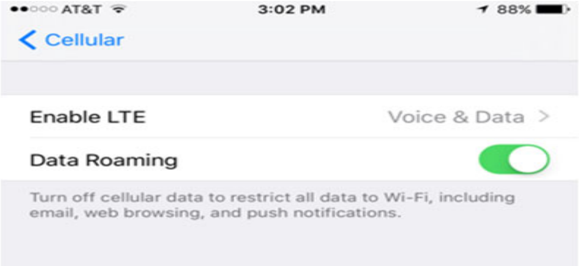
When Wi-Fi Assist is activated, you'll see the cellular data icon in the status bar on your device.

https://support.apple.com/en-us/HT205296



And if we do fall back, then we hide the Wi-Fi indicator, so now the user knows they are not on Wi-Fi anymore. This is something that you get for free, as long as you are using the higher layer APIs, and you should see no difference in your applications apart from a better user experience.

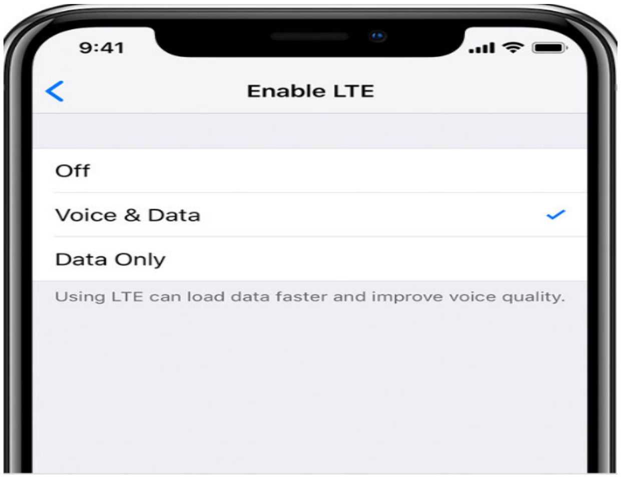
https://developer.apple.com/videos/play/wwdc2015/719/



Enable LTE — Voice & Data >

Data Roaming — ●

Turn off cellular data to restrict all data to Wi-Fi, including email, web browsing, and push notifications.



Enable LTE

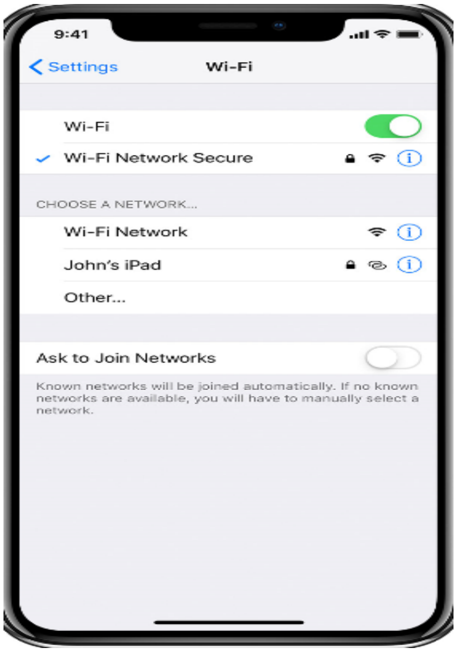
Off

Voice & Data ✓

Data Only

Using LTE can load data faster and improve voice quality.

https://support.apple.com/en-us/HT203124



Wi-Fi

✓ Wi-Fi Network Secure

CHOOSE A NETWORK...

Wi-Fi Network

John's iPad

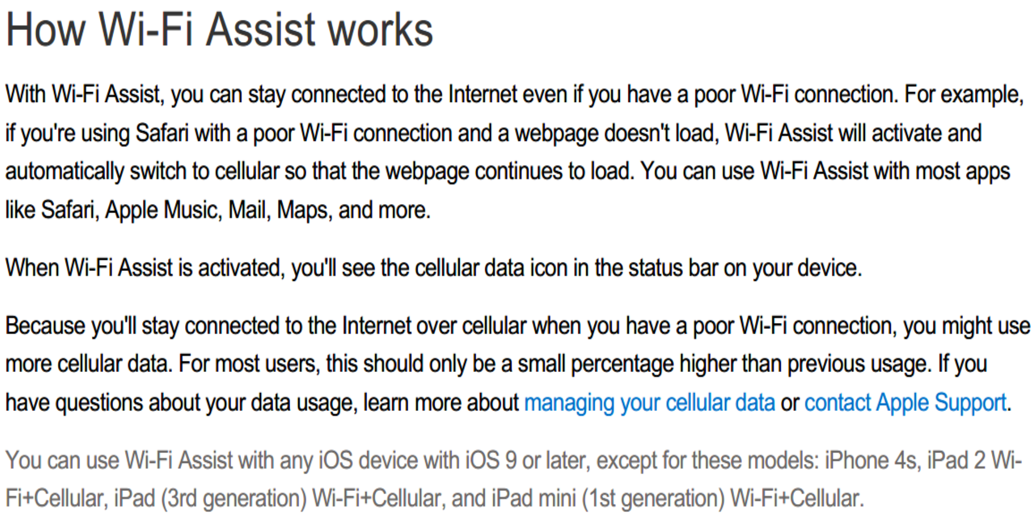
Other...

Ask to Join Networks

Known networks will be joined automatically. If no known networks are available, you will have to manually select a network.

https://support.apple.com/en-us/HT202639

168. Claim 14 of the '557 patent recites "accessing data through the WIFI network in response to an application request from an application executing on a mobile device." The '557 accused products satisfy this limitation. For example, the '557 accused products access data through the Wi-Fi network in response to an application request from an application executing on a mobile device, as shown by the exemplary documentation below.



How Wi-Fi Assist works

With Wi-Fi Assist, you can stay connected to the Internet even if you have a poor Wi-Fi connection. For example, if you're using Safari with a poor Wi-Fi connection and a webpage doesn't load, Wi-Fi Assist will activate and automatically switch to cellular so that the webpage continues to load. You can use Wi-Fi Assist with most apps like Safari, Apple Music, Mail, Maps, and more.

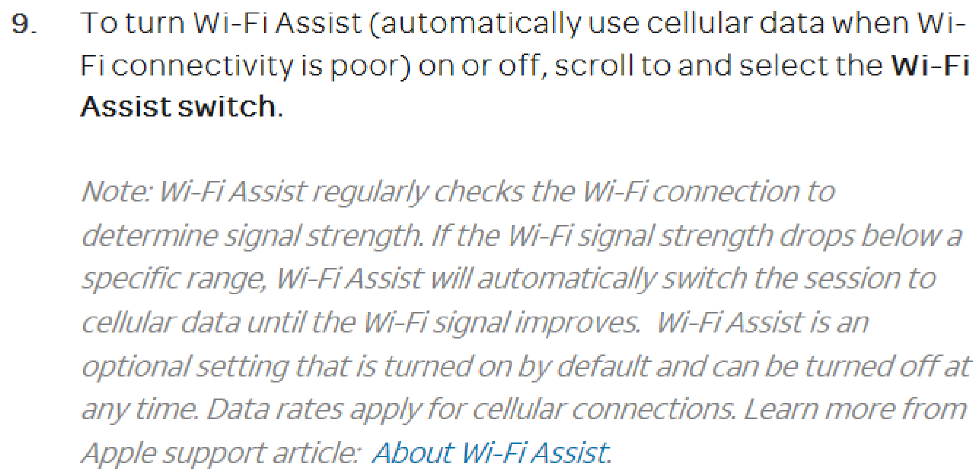
When Wi-Fi Assist is activated, you'll see the cellular data icon in the status bar on your device.

Because you'll stay connected to the Internet over cellular when you have a poor Wi-Fi connection, you might use more cellular data. For most users, this should only be a small percentage higher than previous usage. If you have questions about your data usage, learn more about managing your cellular data or contact Apple Support.

You can use Wi-Fi Assist with any iOS device with iOS 9 or later, except for these models: iPhone 4s, iPad 2 Wi-Fi+Cellular, iPad (3rd generation) Wi-Fi+Cellular, and iPad mini (1st generation) Wi-Fi+Cellular.

https://support.apple.com/en-us/HT205296

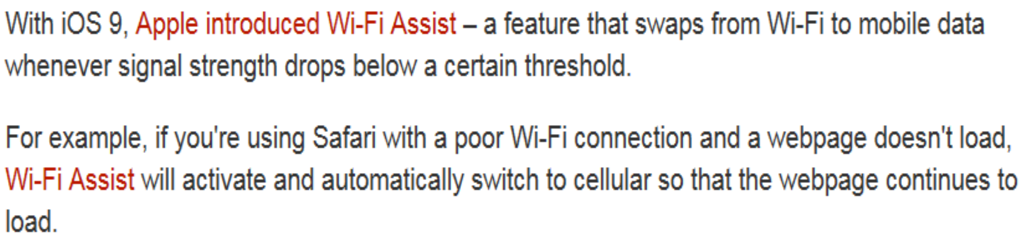
169. Claim 14 of the ’557 patent recites “detecting a first condition indicative of a quality of the WIFI network.” The ’557 accused products satisfy this limitation. For example, the ’557 accused products detect a first condition indicative of a quality of the WIFI network, as shown by the exemplary documentation below.



9. To turn Wi-Fi Assist (automatically use cellular data when Wi-Fi connectivity is poor) on or off, scroll to and select the **Wi-Fi Assist switch**.

   Note: Wi-Fi Assist regularly checks the Wi-Fi connection to determine signal strength. If the Wi-Fi signal strength drops below a specific range, Wi-Fi Assist will automatically switch the session to cellular data until the Wi-Fi signal improves. Wi-Fi Assist is an optional setting that is turned on by default and can be turned off at any time. Data rates apply for cellular connections. Learn more from Apple support article: About Wi-Fi Assist.

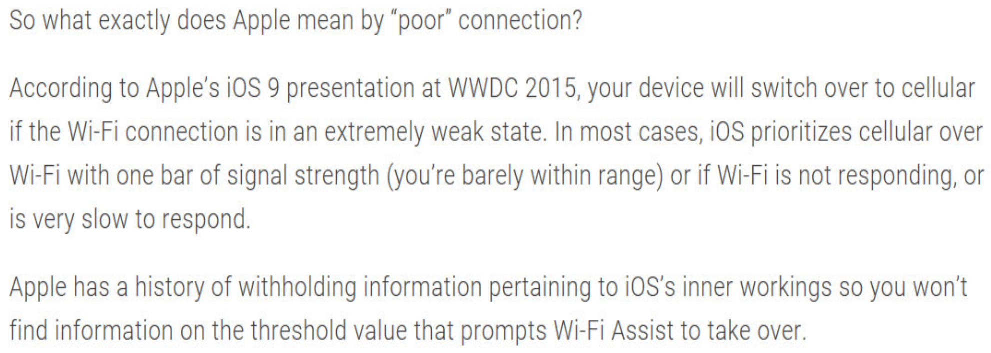
https://www.att.com/devicehowto/tutorial.html#!/stepbystep/id/stepbystep_KM1263746?make=Apple&model=iPhoneSE&gsi=03qouh



With iOS 9, Apple introduced Wi-Fi Assist – a feature that swaps from Wi-Fi to mobile data whenever signal strength drops below a certain threshold.

For example, if you're using Safari with a poor Wi-Fi connection and a webpage doesn't load, Wi-Fi Assist will activate and automatically switch to cellular so that the webpage continues to load.

https://www.express.co.uk/life-style/science-technology/832962/iOS-11-Release-Date-UK-Wi-Fi-Connection



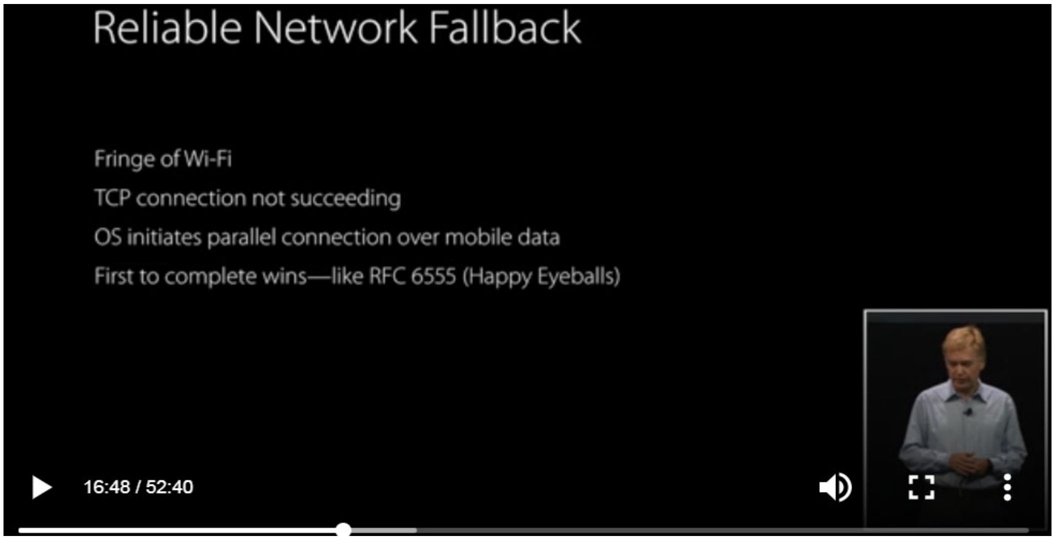
So what exactly does Apple mean by “poor” connection?

According to Apple’s iOS 9 presentation at WWDC 2015, your device will switch over to cellular if the Wi-Fi connection is in an extremely weak state. In most cases, iOS prioritizes cellular over Wi-Fi with one bar of signal strength (you’re barely within range) or if Wi-Fi is not responding, or is very slow to respond.

Apple has a history of withholding information pertaining to iOS’s inner workings so you won’t find information on the threshold value that prompts Wi-Fi Assist to take over.

https://www.idownloadblog.com/2015/09/16/how-to-wi-fi-assist-ios-9/

170. Claim 14 of the ’557 patent recites “detecting in response to a subsequent application request and before or at a time of receiving a response to the subsequent application request, a second condition indicative of a time responsiveness of the WIFI network.” The ’557 accused products satisfy this limitation. For example, the ’557 accused products detect in response to a subsequent application request and before or at a time of receiving a response to the subsequent application request, a second condition indicative of a time responsiveness of a WIFI network, as shown by the exemplary documentation below.



Reliable Network Fallback

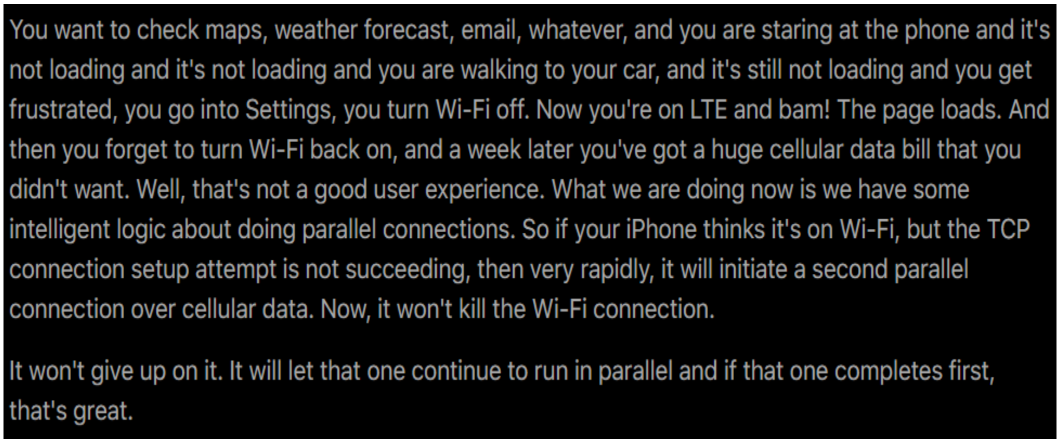
Fringe of Wi-Fi

TCP connection not succeeding

OS initiates parallel connection over mobile data

First to complete wins—like RFC 6555 (Happy Eyeballs)

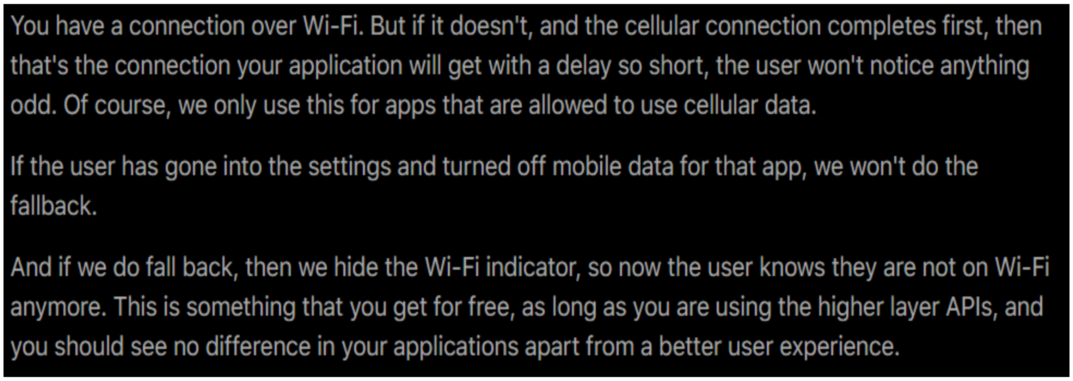
https://developer.apple.com/videos/play/wwdc2015/719/



You want to check maps, weather forecast, email, whatever, and you are staring at the phone and it's not loading and it's not loading and you are walking to your car, and it's still not loading and you get frustrated, you go into Settings, you turn Wi-Fi off. Now you're on LTE and bam! The page loads. And then you forget to turn Wi-Fi back on, and a week later you've got a huge cellular data bill that you didn't want. Well, that's not a good user experience. What we are doing now is we have some intelligent logic about doing parallel connections. So if your iPhone thinks it's on Wi-Fi, but the TCP connection setup attempt is not succeeding, then very rapidly, it will initiate a second parallel connection over cellular data. Now, it won't kill the Wi-Fi connection.

It won't give up on it. It will let that one continue to run in parallel and if that one completes first, that's great.

https://developer.apple.com/videos/play/wwdc2015/719/



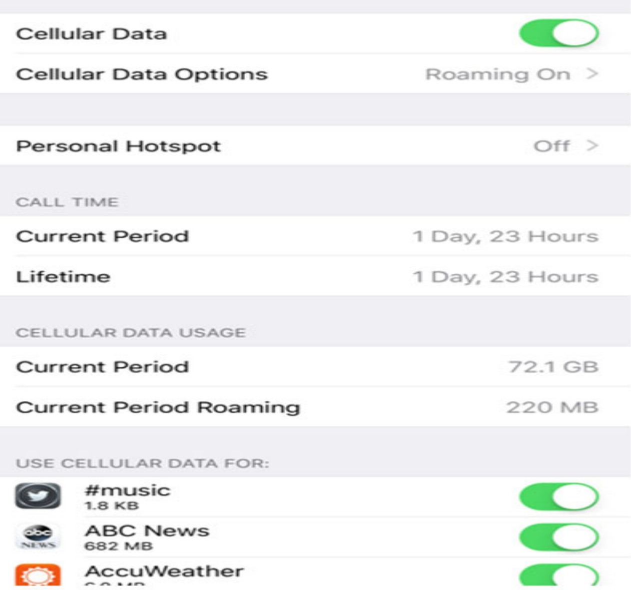
You have a connection over Wi-Fi. But if it doesn't, and the cellular connection completes first, then that's the connection your application will get with a delay so short, the user won't notice anything odd. Of course, we only use this for apps that are allowed to use cellular data.

If the user has gone into the settings and turned off mobile data for that app, we won't do the fallback.

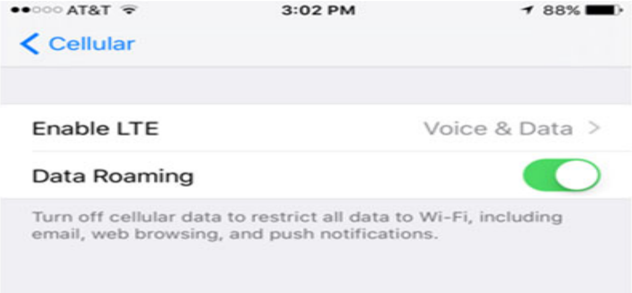
And if we do fall back, then we hide the Wi-Fi indicator, so now the user knows they are not on Wi-Fi anymore. This is something that you get for free, as long as you are using the higher layer APIs, and you should see no difference in your applications apart from a better user experience.

https://developer.apple.com/videos/play/wwdc2015/719/

171. Claim 14 of the ’557 patent recites “evaluating user settings, wherein the user settings include a roaming rule, a connectivity rule, and an application profile of the application.” The ’557 accused products satisfy this limitation. The ’557 accused products evaluate user settings, wherein the user settings include a roaming rule, a connectivity rule, and an application profile of the application, as shown by the exemplary documentation and screenshots below.



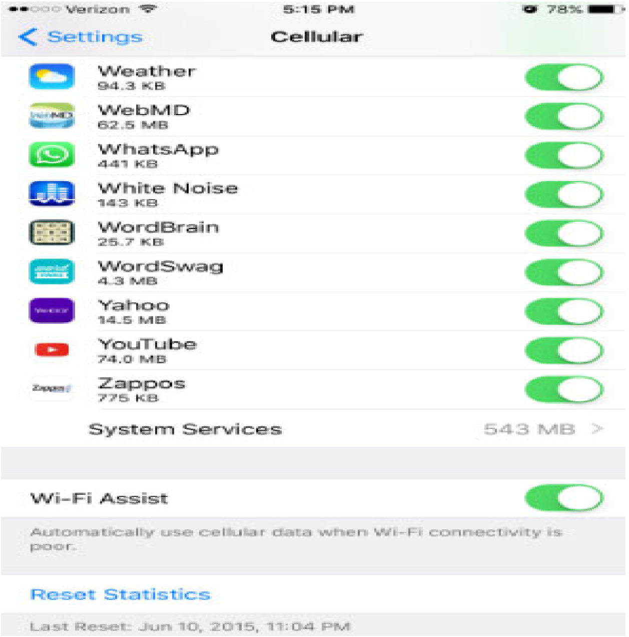
Cellular Data	On
Cellular Data Options	Roaming On >
Personal Hotspot	Off >
CALL TIME	
Current Period	1 Day, 23 Hours
Lifetime	1 Day, 23 Hours
CELLULAR DATA USAGE	
Current Period	72.1 GB
Current Period Roaming	220 MB
USE CELLULAR DATA FOR:	
#music 1.8 KB	On
ABC News 682 MB	On
AccuWeather	On



Enable LTE Voice & Data >

Data Roaming

Turn off cellular data to restrict all data to Wi-Fi, including email, web browsing, and push notifications.



Weather
94.3 KB

WebMD
62.5 MB

WhatsApp
441 KB

White Noise
143 KB

WordBrain
25.7 KB

WordSwag
4.3 MB

Yahoo
14.5 MB

YouTube
74.0 MB

Zappos
775 KB

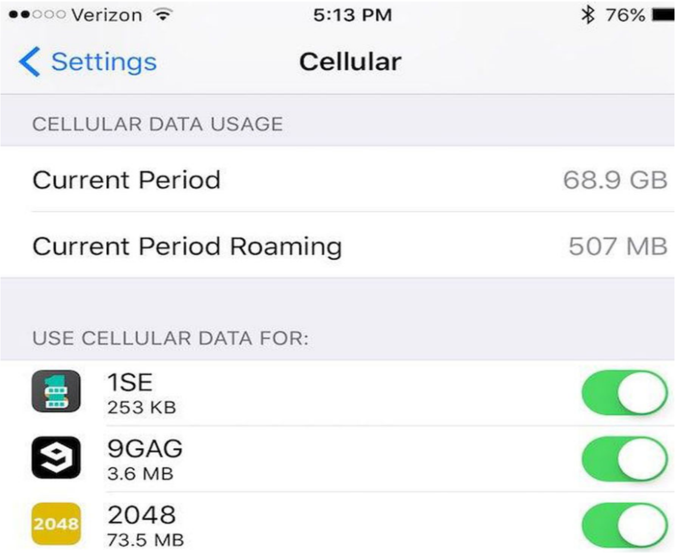
System Services 543 MB >

Wi-Fi Assist

Automatically use cellular data when Wi-Fi connectivity is poor.

Reset Statistics

Last Reset: Jun 10, 2015, 11:04 PM



CELLULAR DATA USAGE

Current Period 68.9 GB

Current Period Roaming 507 MB

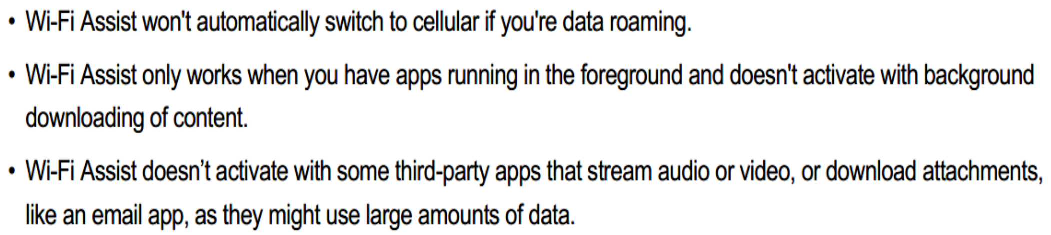
USE CELLULAR DATA FOR:

1SE
253 KB

9GAG
3.6 MB

2048
73.5 MB

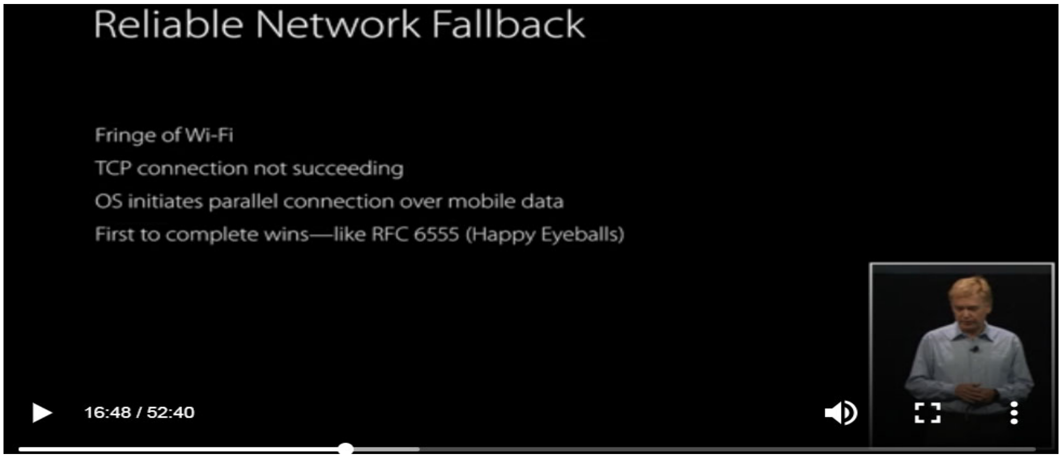
https://www.forbes.com/sites/amitchowdhry/2017/03/24/iphone-tips/#1a70c010316e



- Wi-Fi Assist won't automatically switch to cellular if you're data roaming.
- Wi-Fi Assist only works when you have apps running in the foreground and doesn't activate with background downloading of content.
- Wi-Fi Assist doesn't activate with some third-party apps that stream audio or video, or download attachments, like an email app, as they might use large amounts of data.

https://support.apple.com/en-us/HT205296

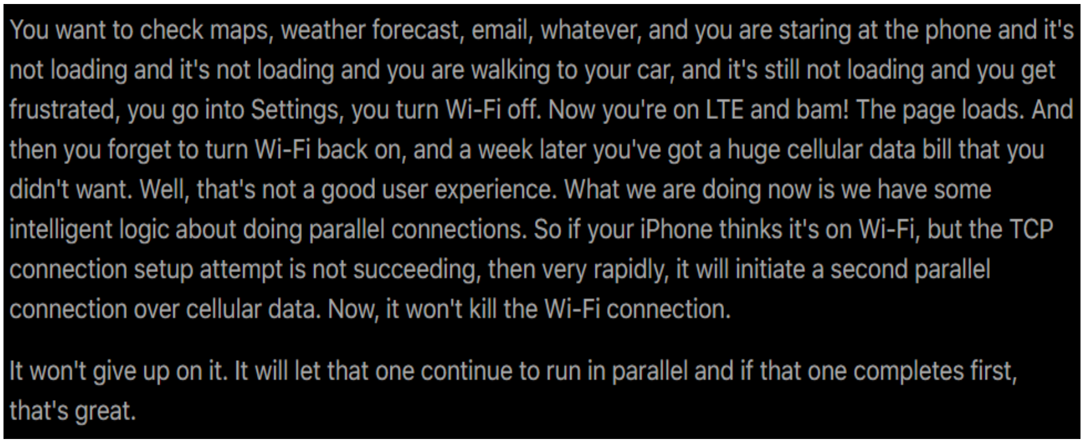
172. Claim 14 of the '557 patent recites "in response to detecting the first condition and the second condition and evaluating the user settings, determining a time responsiveness of the cellular network." The '557 accused products satisfy this limitation. In response to detecting the first condition and evaluating the user settings, the '557 accused products determine a time responsiveness of the cellular network, as shown by the exemplary documentation below.



Reliable Network Fallback

Fringe of Wi-Fi
TCP connection not succeeding
OS initiates parallel connection over mobile data
First to complete wins—like RFC 6555 (Happy Eyeballs)

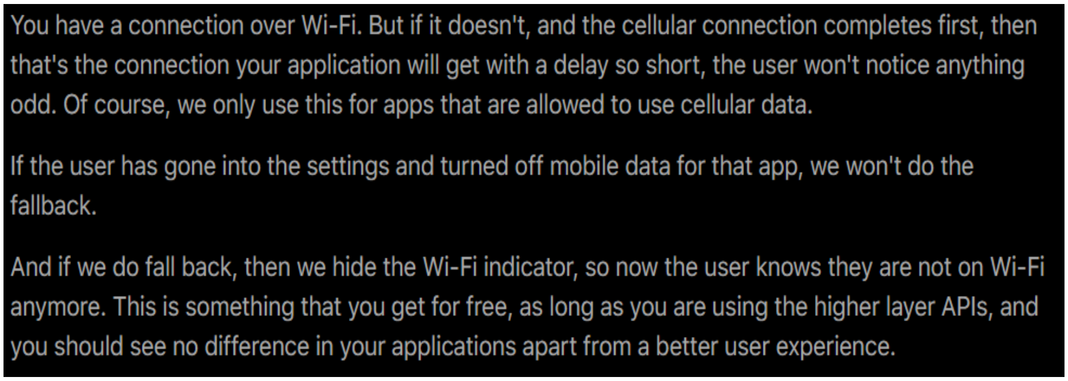
https://developer.apple.com/videos/play/wwdc2015/719/



You want to check maps, weather forecast, email, whatever, and you are staring at the phone and it's not loading and it's not loading and you are walking to your car, and it's still not loading and you get frustrated, you go into Settings, you turn Wi-Fi off. Now you're on LTE and bam! The page loads. And then you forget to turn Wi-Fi back on, and a week later you've got a huge cellular data bill that you didn't want. Well, that's not a good user experience. What we are doing now is we have some intelligent logic about doing parallel connections. So if your iPhone thinks it's on Wi-Fi, but the TCP connection setup attempt is not succeeding, then very rapidly, it will initiate a second parallel connection over cellular data. Now, it won't kill the Wi-Fi connection.

It won't give up on it. It will let that one continue to run in parallel and if that one completes first, that's great.

https://developer.apple.com/videos/play/wwdc2015/719/

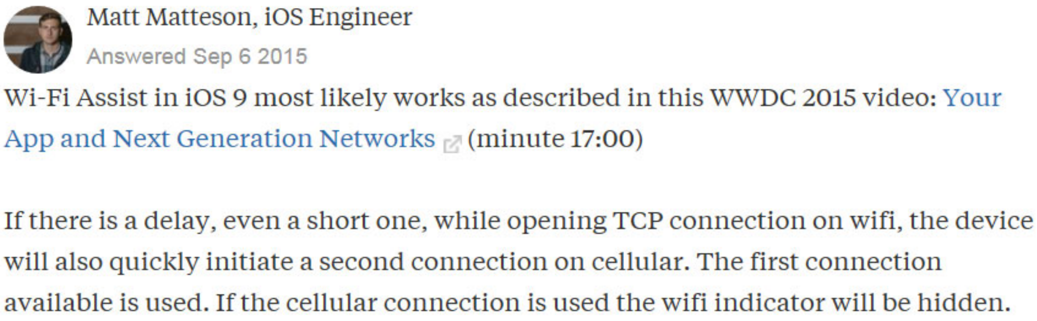


You have a connection over Wi-Fi. But if it doesn't, and the cellular connection completes first, then that's the connection your application will get with a delay so short, the user won't notice anything odd. Of course, we only use this for apps that are allowed to use cellular data.

If the user has gone into the settings and turned off mobile data for that app, we won't do the fallback.

And if we do fall back, then we hide the Wi-Fi indicator, so now the user knows they are not on Wi-Fi anymore. This is something that you get for free, as long as you are using the higher layer APIs, and you should see no difference in your applications apart from a better user experience.

https://developer.apple.com/videos/play/wwdc2015/719/



Matt Matteson, iOS Engineer

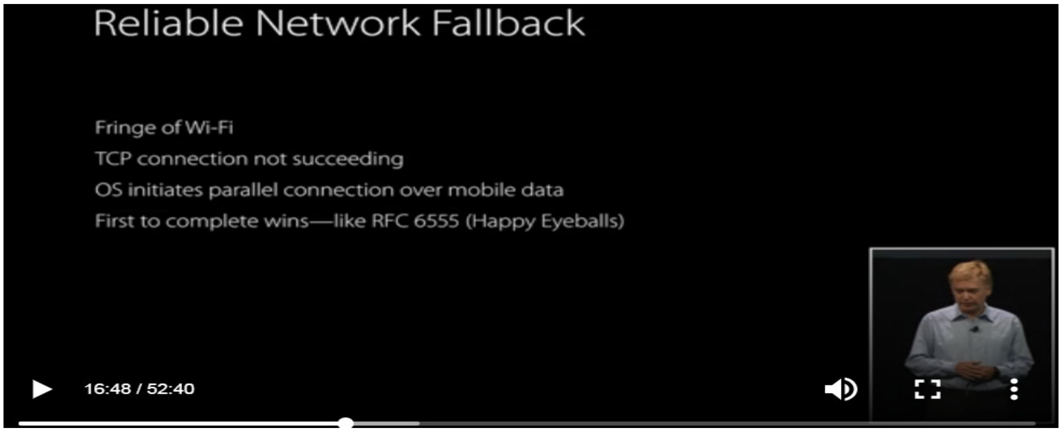
Answered Sep 6 2015

Wi-Fi Assist in iOS 9 most likely works as described in this WWDC 2015 video: Your App and Next Generation Networks (minute 17:00)

If there is a delay, even a short one, while opening TCP connection on wifi, the device will also quickly initiate a second connection on cellular. The first connection available is used. If the cellular connection is used the wifi indicator will be hidden.

https://www.quora.com/How-does-Wi-Fi-Assist-work-on-iOS-9

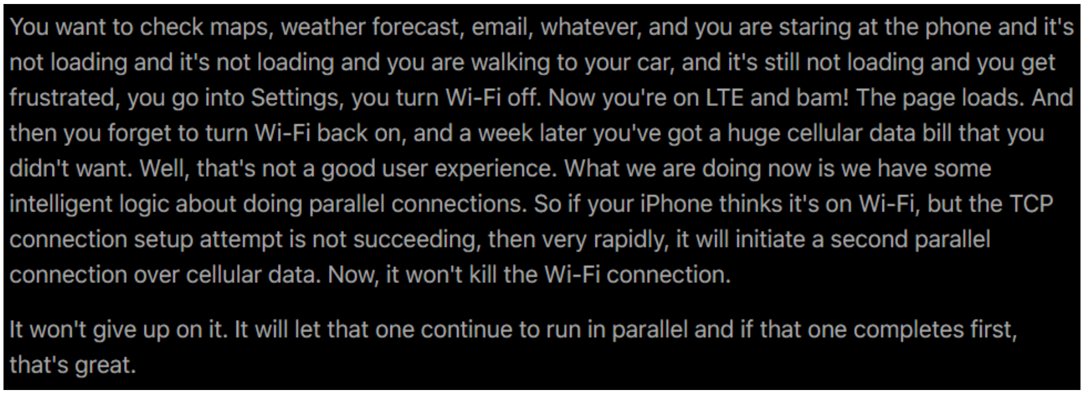
173. Claim 14 of the ’557 patent recites “based on the detected first condition and detected second condition, the evaluated user settings, the time responsiveness of the cellular network, and the application executing on the mobile device, sending the subsequent application request through the cellular network in response to the subsequent application request executing on the mobile device.” The ’557 accused products satisfy this limitation. For example, the ’557 accused products switch from Wi-Fi to cellular using Wi-Fi Assist. Based on the detected first condition and detected second condition, the evaluated user settings, the time responsiveness of the cellular network, and the application executing on the mobile device, the ’557 accused products send the subsequent application request through the cellular network in response to the subsequent application request executing on the mobile device, as shown by the exemplary documentation below.



Reliable Network Fallback

Fringe of Wi-Fi
TCP connection not succeeding
OS initiates parallel connection over mobile data
First to complete wins—like RFC 6555 (Happy Eyeballs)

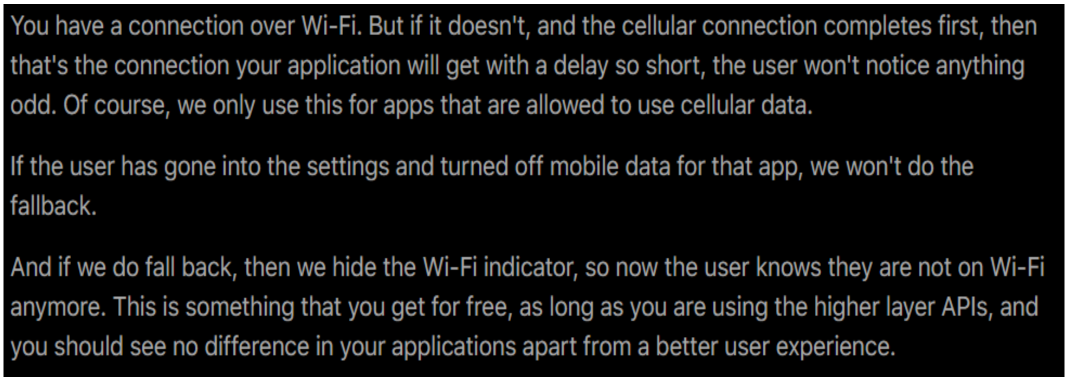
https://developer.apple.com/videos/play/wwdc2015/719/



You want to check maps, weather forecast, email, whatever, and you are staring at the phone and it's not loading and it's not loading and you are walking to your car, and it's still not loading and you get frustrated, you go into Settings, you turn Wi-Fi off. Now you're on LTE and bam! The page loads. And then you forget to turn Wi-Fi back on, and a week later you've got a huge cellular data bill that you didn't want. Well, that's not a good user experience. What we are doing now is we have some intelligent logic about doing parallel connections. So if your iPhone thinks it's on Wi-Fi, but the TCP connection setup attempt is not succeeding, then very rapidly, it will initiate a second parallel connection over cellular data. Now, it won't kill the Wi-Fi connection.

It won't give up on it. It will let that one continue to run in parallel and if that one completes first, that's great.

https://developer.apple.com/videos/play/wwdc2015/719/

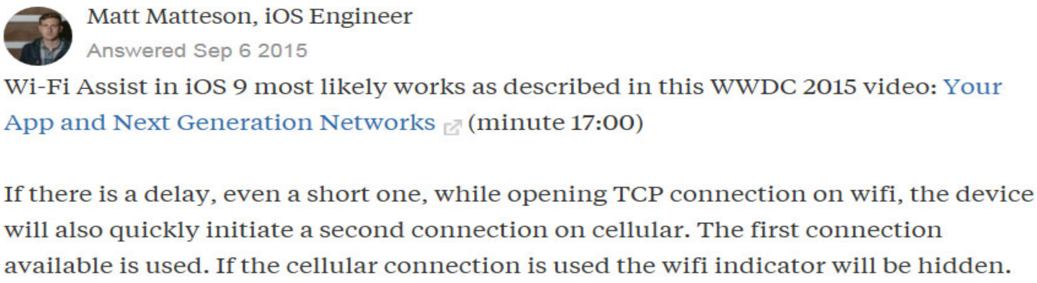


You have a connection over Wi-Fi. But if it doesn't, and the cellular connection completes first, then that's the connection your application will get with a delay so short, the user won't notice anything odd. Of course, we only use this for apps that are allowed to use cellular data.

If the user has gone into the settings and turned off mobile data for that app, we won't do the fallback.

And if we do fall back, then we hide the Wi-Fi indicator, so now the user knows they are not on Wi-Fi anymore. This is something that you get for free, as long as you are using the higher layer APIs, and you should see no difference in your applications apart from a better user experience.

https://developer.apple.com/videos/play/wwdc2015/719/

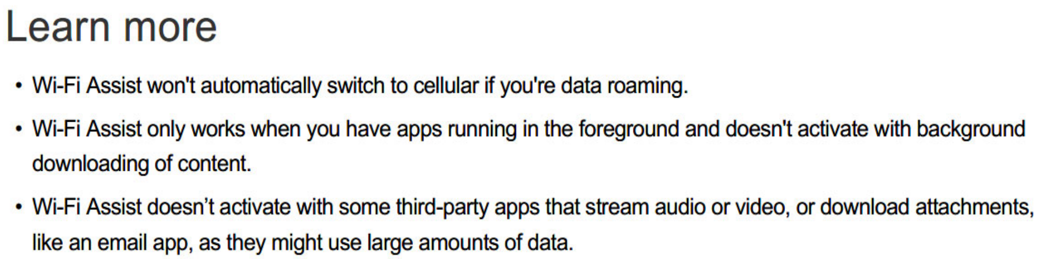


Matt Matteson, iOS Engineer
Answered Sep 6 2015

Wi-Fi Assist in iOS 9 most likely works as described in this WWDC 2015 video: Your App and Next Generation Networks (minute 17:00)

If there is a delay, even a short one, while opening TCP connection on wifi, the device will also quickly initiate a second connection on cellular. The first connection available is used. If the cellular connection is used the wifi indicator will be hidden.

https://www.quora.com/How-does-Wi-Fi-Assist-work-on-iOS-9



Learn more

- Wi-Fi Assist won't automatically switch to cellular if you're data roaming.
- Wi-Fi Assist only works when you have apps running in the foreground and doesn't activate with background downloading of content.
- Wi-Fi Assist doesn't activate with some third-party apps that stream audio or video, or download attachments, like an email app, as they might use large amounts of data.

https://support.apple.com/en-us/HT205296

174. Thus, as illustrated above, the ’557 accused products directly infringe one or more claims of the ’557 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the ’557 patent.

175. Apple indirectly infringes the ’557 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple’s customers and end-users, in this district and elsewhere in the United States. For example, Apple’s customers and end-users directly infringe through their use of the inventions claimed in the ’557 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’557 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the ’557 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products’ capability in an infringing manner,

including through the use of Wi-Fi Assist. https://support.apple.com/en-us/HT205296; https://developer.apple.com/videos/play/wwdc2015/719/. As a result of Apple's inducement, Apple's customers and end-users use the Apple products in the way Apple intends and directly infringe the '557 patent. Apple performs these affirmative acts with knowledge of the '557 patent and with the intent, or willful blindness, that the induced acts directly infringe the '557 patent. Apple has had knowledge and notice of the '557 patent at least as of the filing of this complaint.

176. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

COUNT VIII: PATENT INFRINGEMENT OF THE '476 PATENT

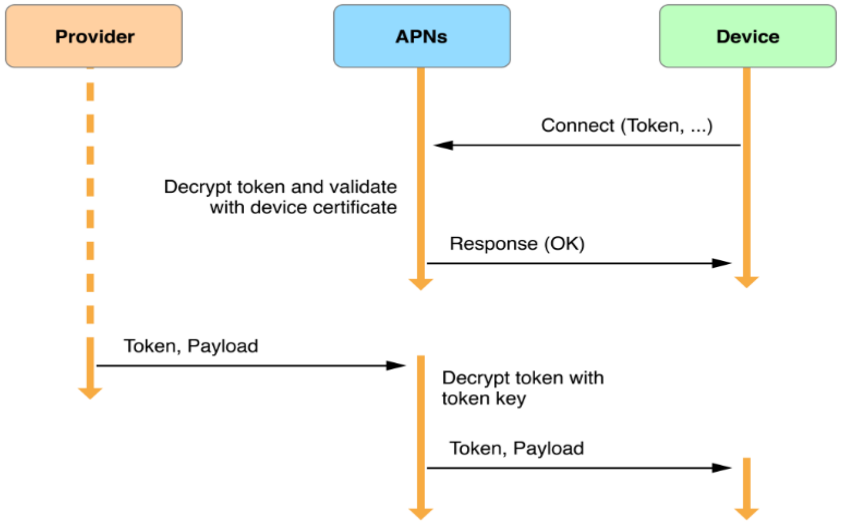
177. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

178. Apple infringes (literally and/or under the Doctrine of Equivalents) the '476 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the '476 patent including at least the '476 accused products noted above. In addition, Apple derives revenue from the activities relating to the '476 accused products.

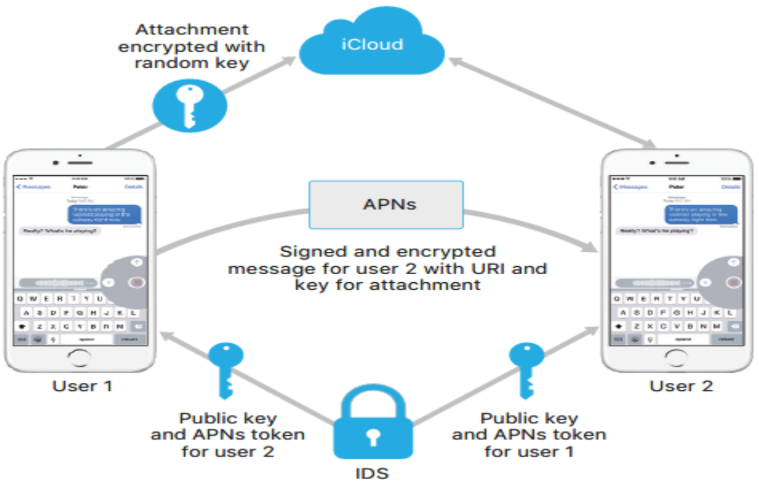
179. For example and as shown below, the '476 accused products infringe at least claims 23 and 33 of the '476 patent. For example, per claim 23, to the extent the preamble is limiting, Apple's server satisfies a "server for processing a transaction, the server having a processor configured to:" As shown below by the exemplary evidence, Apple's server processes a transaction between a first computer, such as an iPhone, iPad, or mac, and a second computer,

such as a provider or another iPhone, iPad, or mac, as shown by the exemplary documentation below.

Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

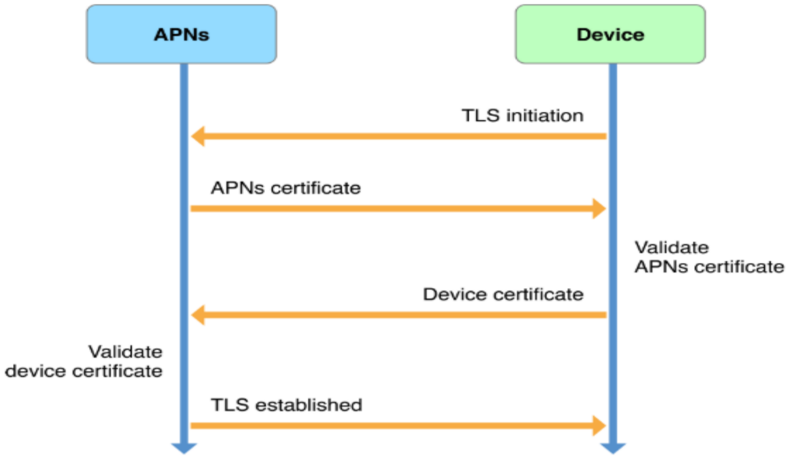

Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
IDS
Public key and APNs token for user 1

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

180. Claim 23 of the ’476 patent recites “receive a username and a password from a first computer; authenticate the username and the password with a user database;” Apple’s server satisfies this limitation. Apple’s server receives a username and a password from a first

computer, such as an Apple iPhone, iPad, or mac, and authenticate the username and the password with a user database, as shown by the exemplary documentation below.

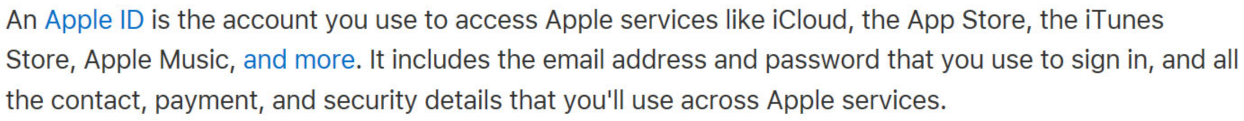
Figure 6-5 Establishing connection trust between a device and APNs



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

```
default 20:40:27.554127 -0400  apsd     2019-03-16 20:40:27 -0400 apsd[85]: <APSCourier:
0x11bd0d530>: Sending filter message for: token type systemToken, token: <6dc220e2
a40b027a d01c77ce 7df3f958 fdbb9ec5 bf218587 fc444b14 e2e9c4b1>, enabled hashes {
    <307ce9d8 26877944 a0cda5fb 151965fe ad398f9f> = "com.spotify.client";
    <bb5ddb5d 1674248f 6a18c0b9 ec4fec0b a9477d60> = "com.apple.maps.icloud";
    <63a82ad5 e27b04a3 5cefda08 e3773f25 adf3b8d6> = "com.me.keyvalueservice";
    <0d77daf0 96ceb132 655f3a29 12118ce8 609026a3> = "us.parkmobile.ParkMobile";
    <fa3049fc 04e3afe0 752e2938 8f086616 16faac92> = "com.google.hangouts.voip";
    <a703ff59 e0d4dfe6 1fba112c 4f744c38 5fae93ce> = "com.facebook.Messenger.voip";
    <6e2f732d d6872e1f 58f193dd 18638d18 0b78c5a0> = "com.ubercab.UberEats";
    <a1d17919 7ab41a91 59e473b6 0c8ddcf9 4ed8e840> = "jp.naver.line";
    <e82aba2c 4933a627 094d2ed6 4f60f3f6 2e288afa> = "com.amtrak.rider";
    <e7e89581 8ca86c28 863dce40 2c5092b9 8ac35b20> = "com.skype.skype";
    <f9b12762 c0dcf6f0 da651a65<…>
default 20:40:27.554586 -0400  apsd     2019-03-16 20:40:27 -0400 apsd[85]: Stopped indexing
hashes at index 38 {token: <6dc220e2 a40b027a d01c77ce 7df3f958 fdbb9ec5 bf218587
fc444b14 e2e9c4b1>, enabledHashes.count: 110, ignoredHashes.count: 1,
opportunisticHashes.count: 95, pausedHashes.count: 0}
```

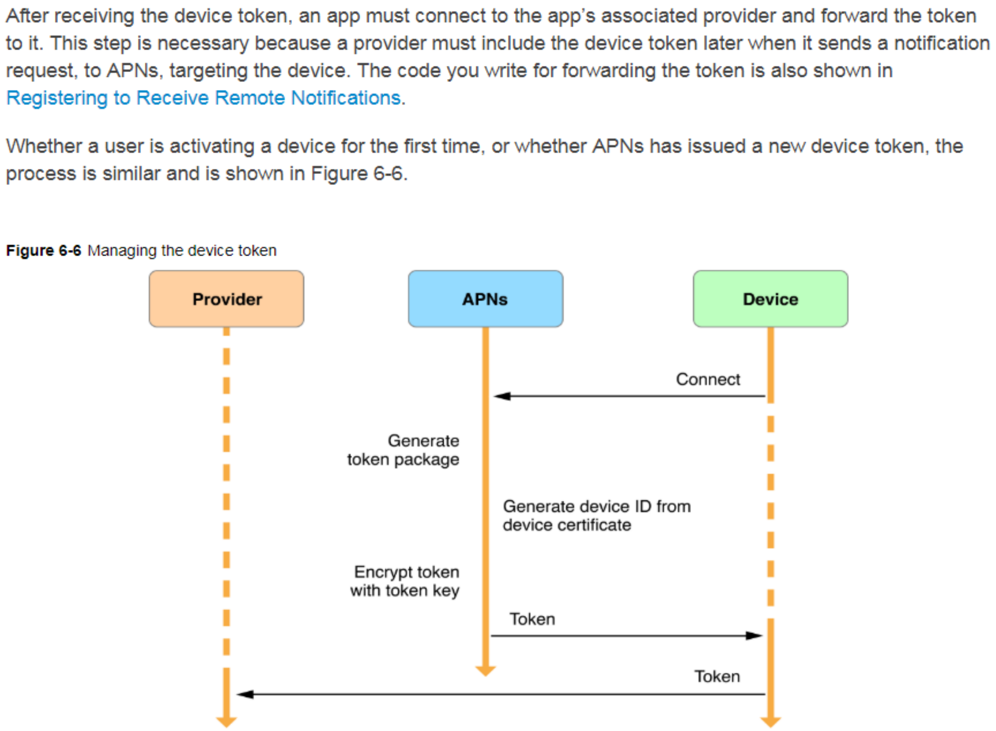
iPhone log captured March 16, 2019



An Apple ID is the account you use to access Apple services like iCloud, the App Store, the iTunes Store, Apple Music, and more. It includes the email address and password that you use to sign in, and all the contact, payment, and security details that you'll use across Apple services.

https://support.apple.com/en-us/HT204316

181. Claim 23 of the ’476 patent recites “issue a token for the first computer after authenticating the username and the password, wherein a first point-to-point security association is negotiated with the first computer and a second point-to-point security association is negotiated with a second computer.” Apple’s server satisfies this limitation. Apple’s server issues a token for a first computer after authenticating the username and the password. A first point-to-point security association is negotiated with the first computer and a second point-to-point security association is negotiated with a second computer, as shown by the exemplary documentation below.



After receiving the device token, an app must connect to the app’s associated provider and forward the token to it. This step is necessary because a provider must include the device token later when it sends a notification request, to APNs, targeting the device. The code you write for forwarding the token is also shown in Registering to Receive Remote Notifications.

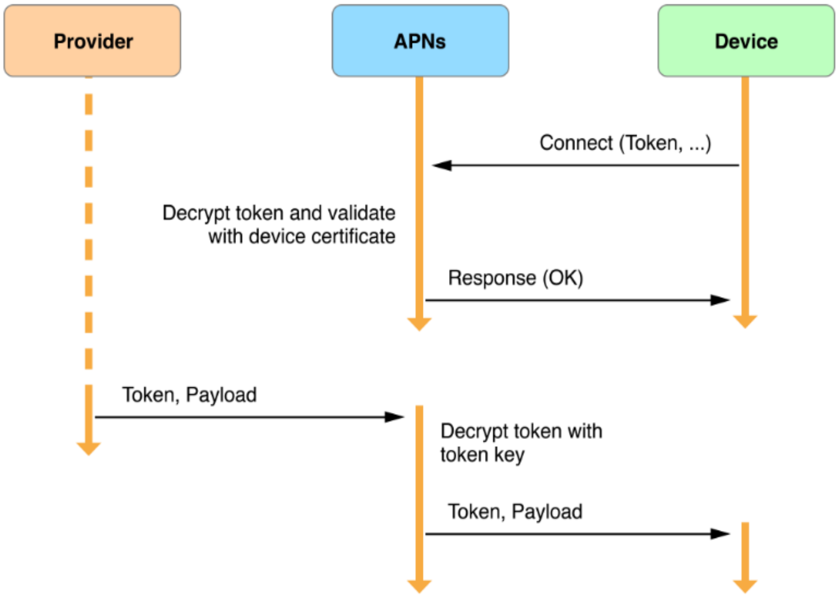
Whether a user is activating a device for the first time, or whether APNs has issued a new device token, the process is similar and is shown in Figure 6-6.

Figure 6-6 Managing the device token

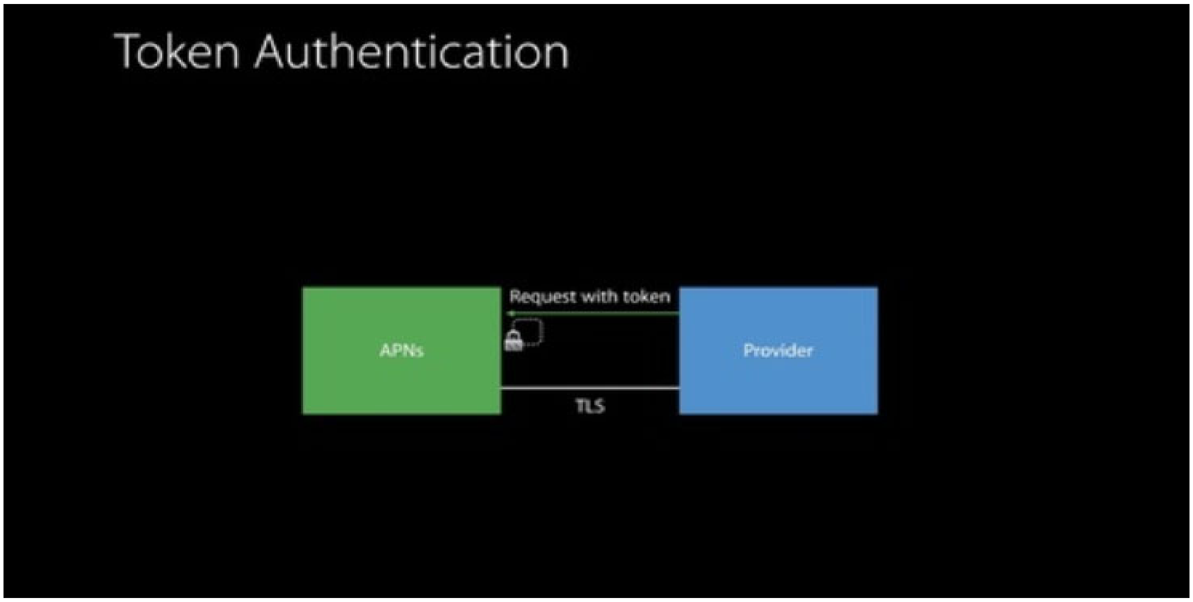
Provider
APNs
Device
Connect
Generate token package
Generate device ID from device certificate
Encrypt token with token key
Token
Token

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

Figure 6-7 Remote notification path from provider to device

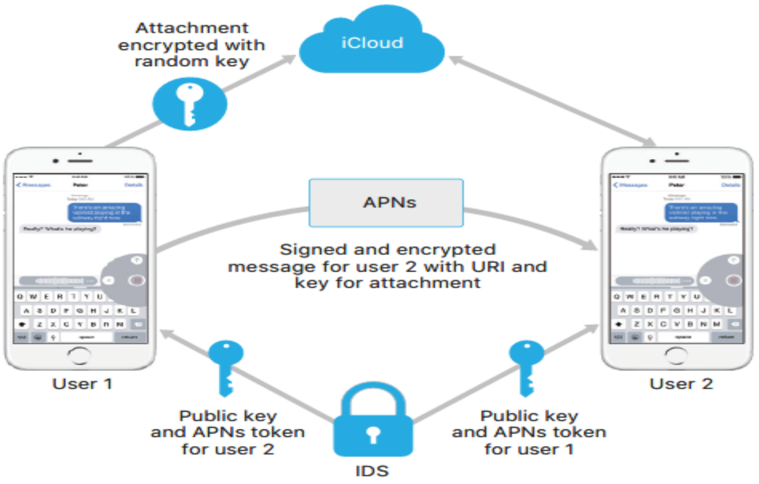


https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1



Token Authentication

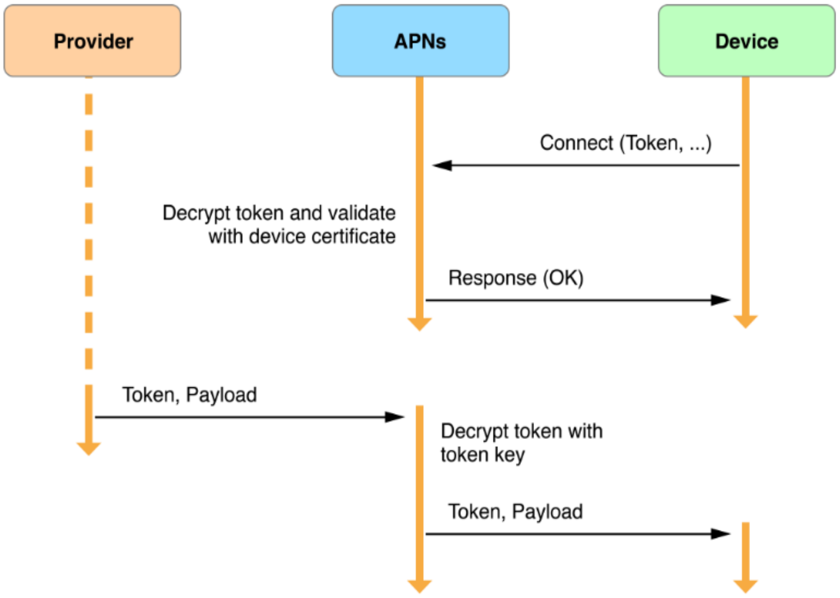
https://developer.apple.com/videos/wwdc2016/


Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS

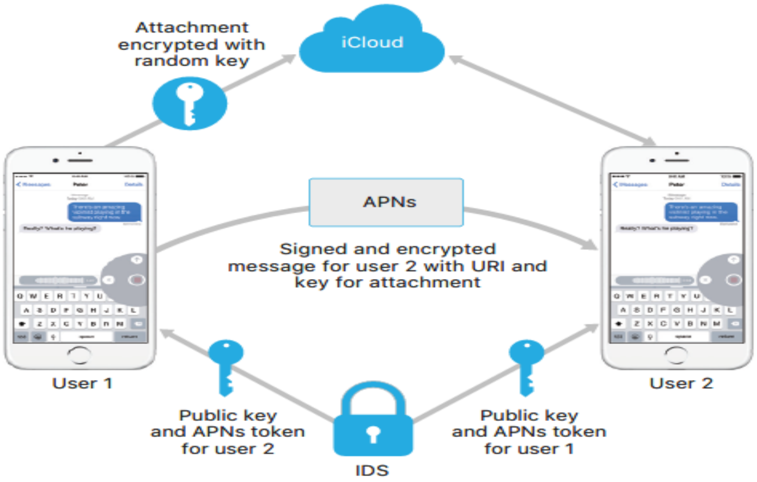
https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

182. Claim 23 of the ’476 patent recites “receive a transaction message from the second computer, the transaction message comprising control data and payload data, wherein the control data includes information that provides authentication of a source of the transaction and transaction routing information, wherein the information includes the token.” Apple’s server satisfies this limitation. Apple’s server receives a transaction message from a second computer, such as a provider, which comprises control data and payload data, wherein the control data includes information that provides authentication of the transaction and transaction routing information, wherein the information includes the token, as shown by the exemplary documentation below.

Figure 6-7 Remote notification path from provider to device



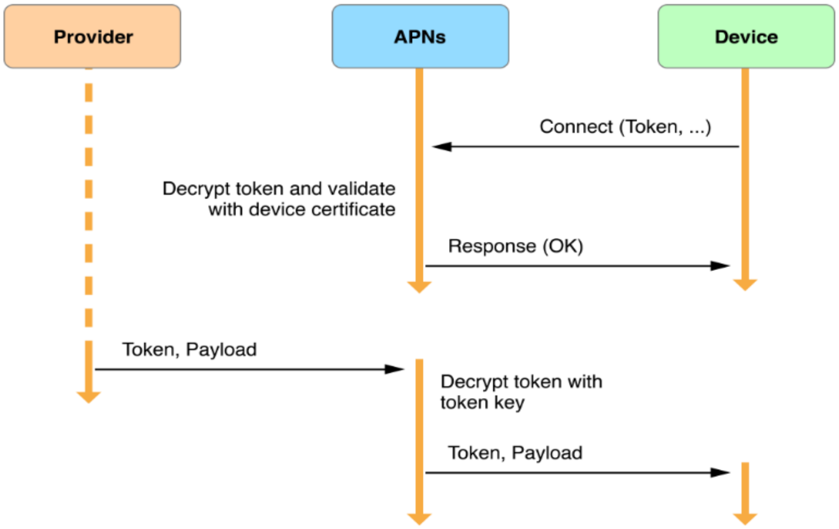
https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1


Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS

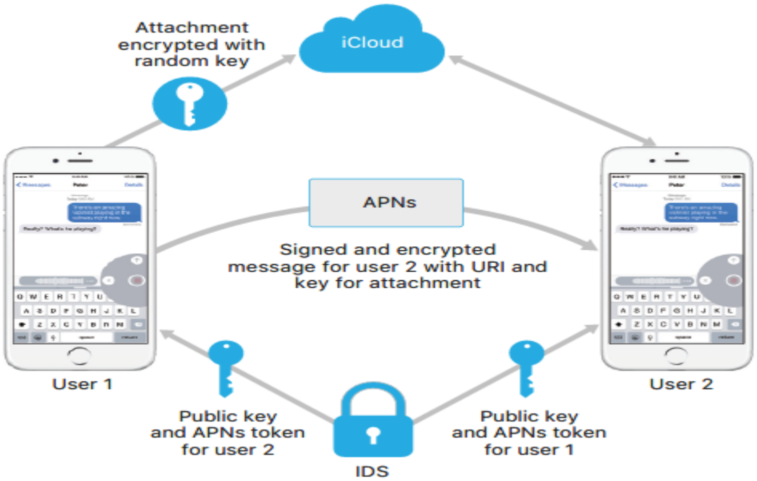
https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

183. Claim 23 of the ’476 patent recites “transmit the payload data to the first computer based on the transaction routing information.” Apple’s server satisfies this limitation. Apple’s server transmits payload data to the first computer based on the transaction routing information, as shown by the exemplary documentation below.

Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1


Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

184. Claim 33 of the ’476 patent recites a “first computer having a processor configured to:” To the extent the preamble is limiting, the ’476 accused products (e.g., iPhones, iPads, and macs) satisfy a first computer having a processor.

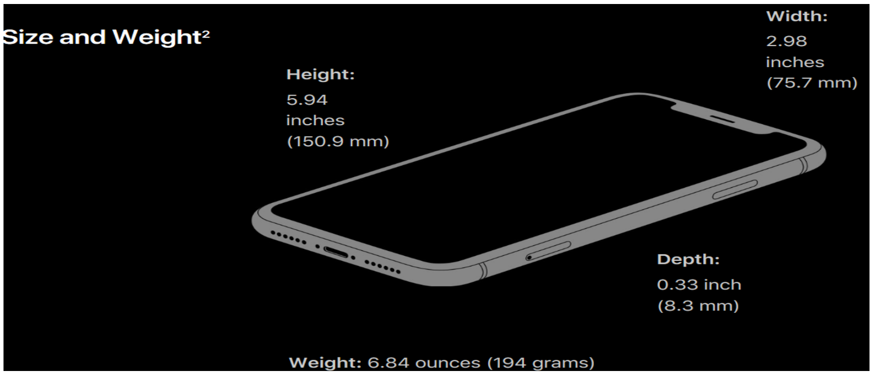
Chip

A12

A12 Bionic chip

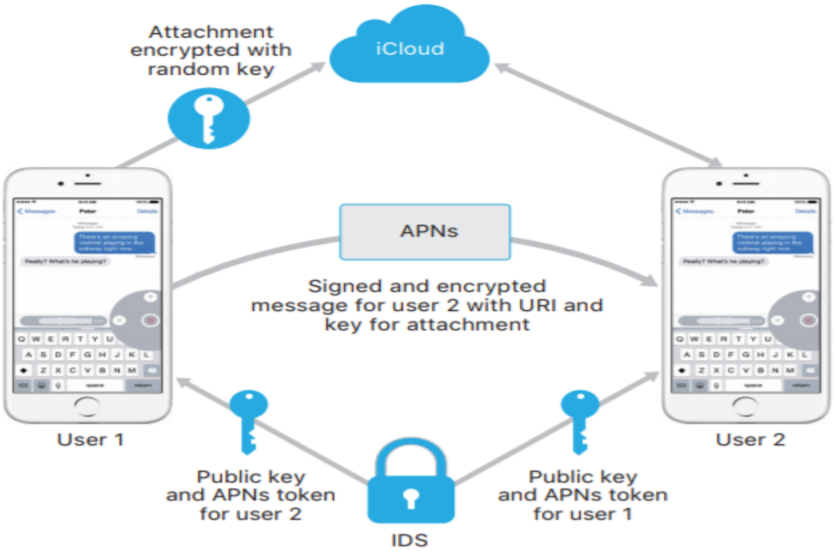
Next-generation Neural Engine

https://www.apple.com/iphone-xr/specs/

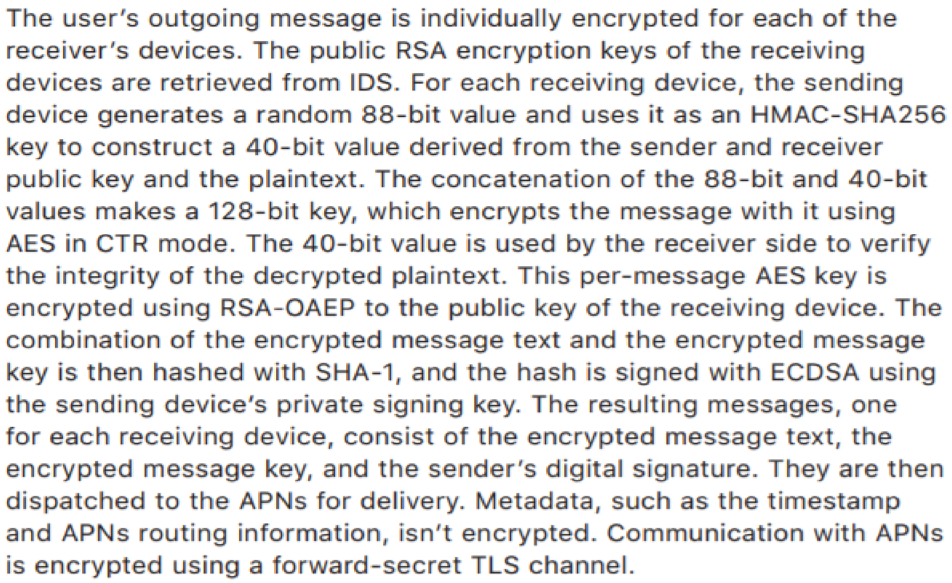


https://www.apple.com/iphone-xr/specs/

185. Claim 33 of the '476 patent recites "encrypt first data of a first data path in a transaction using a first security association, wherein the first data path is through an intermediary server that provides connectivity between the first computer and a second computer, and wherein the first security association is not known to the intermediary server." The '476 accused products satisfy this limitation. The '476 accused products encrypt first data of a first data path in a transaction using a first security association, wherein the first data path is through an intermediary server that provides connectivity between the first computer and a second computer, and wherein the first security association is not known to the intermediary server, as shown in the exemplary documentation below.

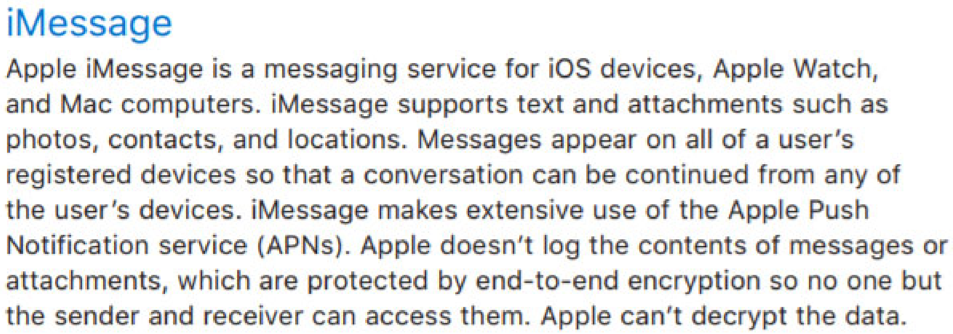

Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf



The user's outgoing message is individually encrypted for each of the receiver's devices. The public RSA encryption keys of the receiving devices are retrieved from IDS. For each receiving device, the sending device generates a random 88-bit value and uses it as an HMAC-SHA256 key to construct a 40-bit value derived from the sender and receiver public key and the plaintext. The concatenation of the 88-bit and 40-bit values makes a 128-bit key, which encrypts the message with it using AES in CTR mode. The 40-bit value is used by the receiver side to verify the integrity of the decrypted plaintext. This per-message AES key is encrypted using RSA-OAEP to the public key of the receiving device. The combination of the encrypted message text and the encrypted message key is then hashed with SHA-1, and the hash is signed with ECDSA using the sending device's private signing key. The resulting messages, one for each receiving device, consist of the encrypted message text, the encrypted message key, and the sender's digital signature. They are then dispatched to the APNs for delivery. Metadata, such as the timestamp and APNs routing information, isn't encrypted. Communication with APNs is encrypted using a forward-secret TLS channel.

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

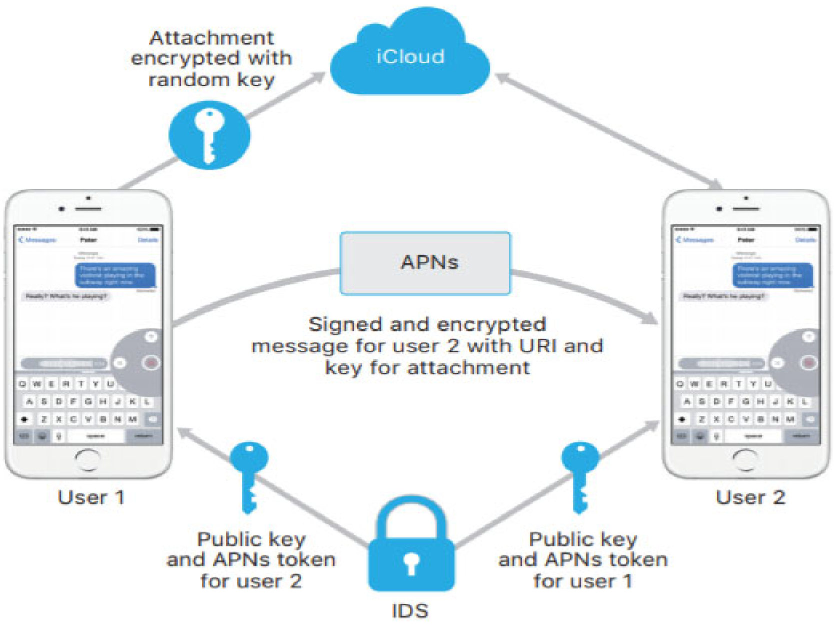


iMessage

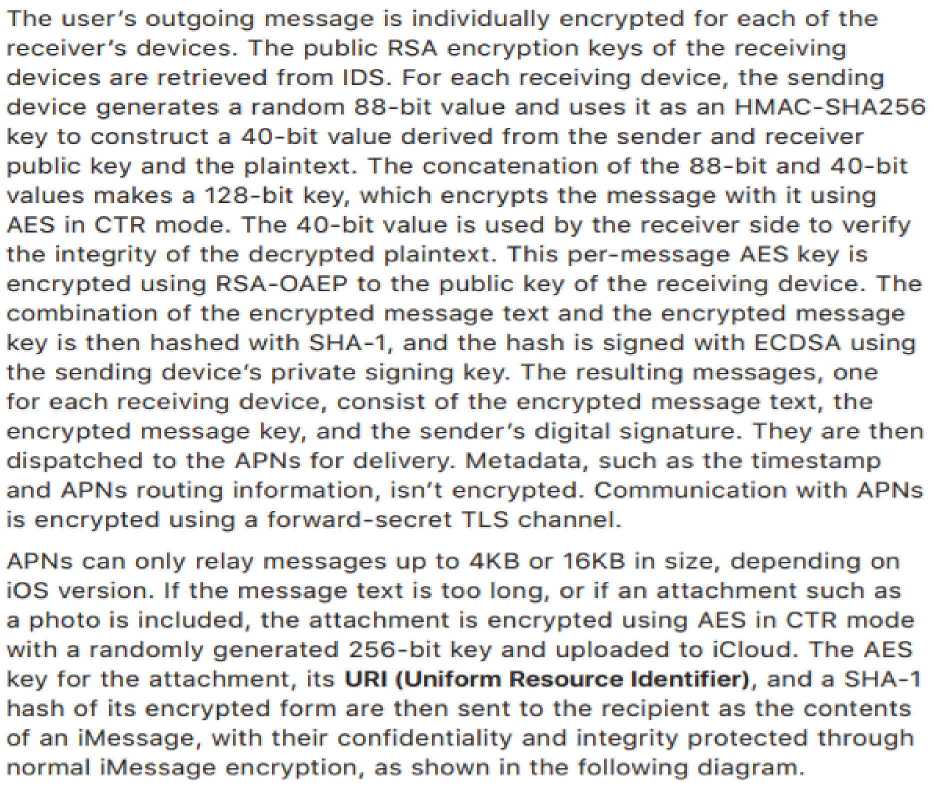
Apple iMessage is a messaging service for iOS devices, Apple Watch, and Mac computers. iMessage supports text and attachments such as photos, contacts, and locations. Messages appear on all of a user's registered devices so that a conversation can be continued from any of the user's devices. iMessage makes extensive use of the Apple Push Notification service (APNs). Apple doesn't log the contents of messages or attachments, which are protected by end-to-end encryption so no one but the sender and receiver can access them. Apple can't decrypt the data.

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

186. Claim 33 of the '476 patent recites "wherein the transaction comprises a transaction message that includes control data and payload data." The '476 accused products satisfy this limitation. For example, the '476 accused products send control data and payload data.


Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf



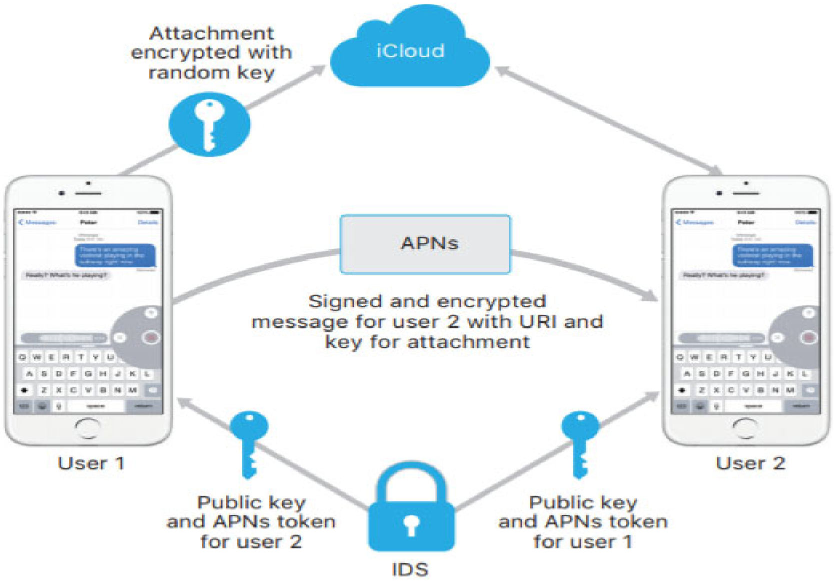
The user's outgoing message is individually encrypted for each of the receiver's devices. The public RSA encryption keys of the receiving devices are retrieved from IDS. For each receiving device, the sending device generates a random 88-bit value and uses it as an HMAC-SHA256 key to construct a 40-bit value derived from the sender and receiver public key and the plaintext. The concatenation of the 88-bit and 40-bit values makes a 128-bit key, which encrypts the message with it using AES in CTR mode. The 40-bit value is used by the receiver side to verify the integrity of the decrypted plaintext. This per-message AES key is encrypted using RSA-OAEP to the public key of the receiving device. The combination of the encrypted message text and the encrypted message key is then hashed with SHA-1, and the hash is signed with ECDSA using the sending device's private signing key. The resulting messages, one for each receiving device, consist of the encrypted message text, the encrypted message key, and the sender's digital signature. They are then dispatched to the APNs for delivery. Metadata, such as the timestamp and APNs routing information, isn't encrypted. Communication with APNs is encrypted using a forward-secret TLS channel.

APNs can only relay messages up to 4KB or 16KB in size, depending on iOS version. If the message text is too long, or if an attachment such as a photo is included, the attachment is encrypted using AES in CTR mode with a randomly generated 256-bit key and uploaded to iCloud. The AES key for the attachment, its **URI (Uniform Resource Identifier)**, and a SHA-1 hash of its encrypted form are then sent to the recipient as the contents of an iMessage, with their confidentiality and integrity protected through normal iMessage encryption, as shown in the following diagram.

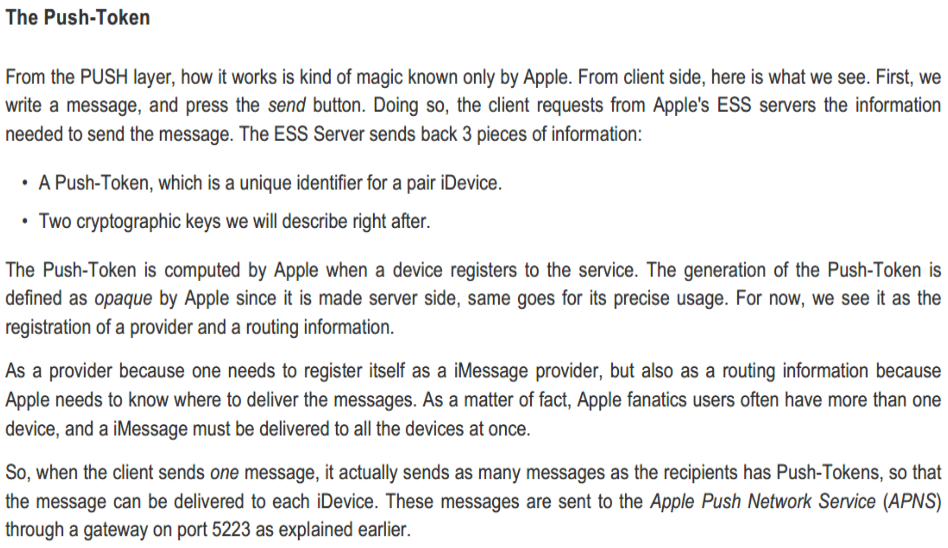
https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

187. Claim 33 of the '476 patent recites "transmit the control data to the intermediary server, wherein the control data includes a token associated with the intermediary server and the

token provides transaction routing information.” The ’476 accused products satisfy this limitation. For example,the ’476 accused products transmit control data to the intermediary server, wherein the control data includes a token associated with the intermediary server and the token provides transaction routing information, as shown by the exemplary documentation below.


Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf



The Push-Token

From the PUSH layer, how it works is kind of magic known only by Apple. From client side, here is what we see. First, we write a message, and press the *send* button. Doing so, the client requests from Apple's ESS servers the information needed to send the message. The ESS Server sends back 3 pieces of information:

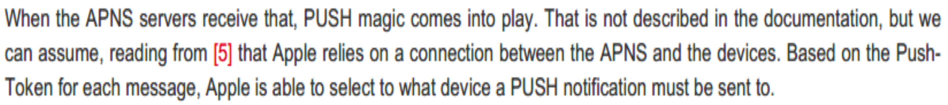
- A Push-Token, which is a unique identifier for a pair iDevice.
- Two cryptographic keys we will describe right after.

The Push-Token is computed by Apple when a device registers to the service. The generation of the Push-Token is defined as *opaque* by Apple since it is made server side, same goes for its precise usage. For now, we see it as the registration of a provider and a routing information.

As a provider because one needs to register itself as a iMessage provider, but also as a routing information because Apple needs to know where to deliver the messages. As a matter of fact, Apple fanatics users often have more than one device, and a iMessage must be delivered to all the devices at once.

So, when the client sends *one* message, it actually sends as many messages as the recipients has Push-Tokens, so that the message can be delivered to each iDevice. These messages are sent to the *Apple Push Network Service* (*APNS*) through a gateway on port 5223 as explained earlier.

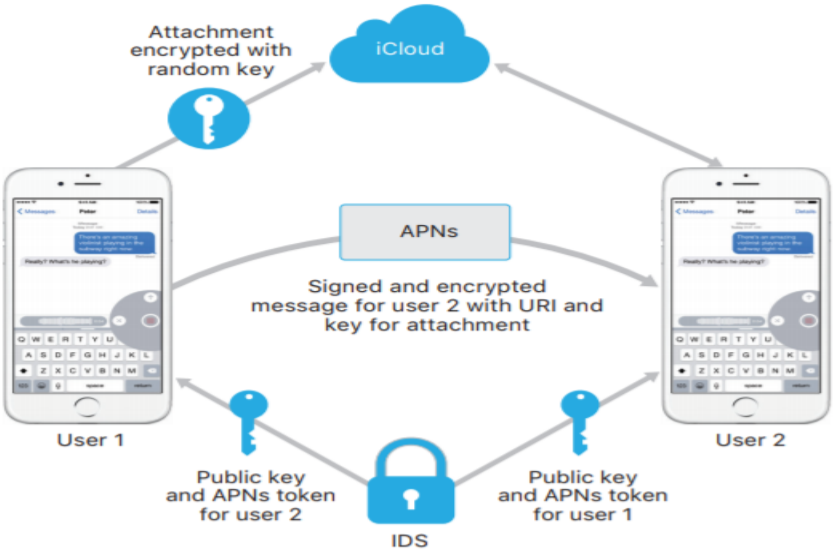
….



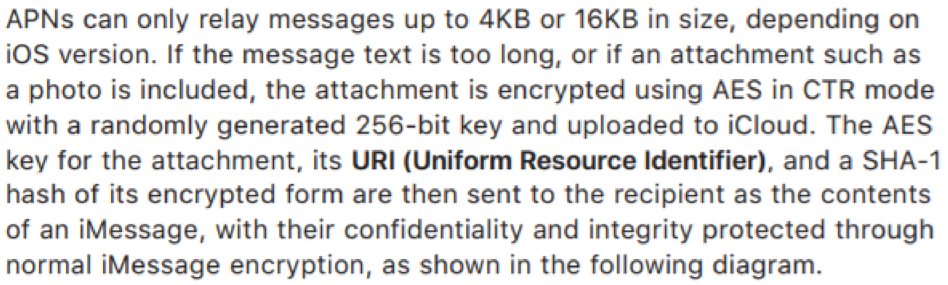
When the APNS servers receive that, PUSH magic comes into play. That is not described in the documentation, but we can assume, reading from [5] that Apple relies on a connection between the APNS and the devices. Based on the Push-Token for each message, Apple is able to select to what device a PUSH notification must be sent to.

https://blog.quarkslab.com/imessage-privacy.html

188. Claim 33 of the ’476 patent recites “encrypt second data of a second data path using a second security association, wherein the second data path is distinct from the intermediary server.” The ’476 accused products satisfy this limitation. The ’476 accused products encrypt second data of a second data path using a second security association, and send the second data through a second data path distinct from the intermediary server, as shown by the exemplary documentation below.


Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf



APNs can only relay messages up to 4KB or 16KB in size, depending on iOS version. If the message text is too long, or if an attachment such as a photo is included, the attachment is encrypted using AES in CTR mode with a randomly generated 256-bit key and uploaded to iCloud. The AES key for the attachment, its **URI (Uniform Resource Identifier)**, and a SHA-1 hash of its encrypted form are then sent to the recipient as the contents of an iMessage, with their confidentiality and integrity protected through normal iMessage encryption, as shown in the following diagram.

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

189. Thus, as illustrated above, the ’476 accused products directly infringe one or more claims of the ’476 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the ’476 patent.

190. Apple indirectly infringes the ’476 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), by inducing infringement by others, such as Apple’s customers and end-users, in this district and elsewhere in the United States. For example, Apple’s customers and end-users directly infringe through their use of the inventions claimed in the ’476 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’476 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the ’476 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products’ capability in an infringing manner, including through the use of Apple’s Push Notification service and iMessage. https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1; https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf. As a result of Apple’s inducement, Apple’s customers and end-users use the Apple products in the way Apple intends

and directly infringe the ’476 patent. Apple performs these affirmative acts with knowledge of the ’476 patent and with the intent, or willful blindness, that the induced acts directly infringe the ’476 patent. Apple has had knowledge and notice of the ’476 patent at least as of the filing of this complaint.

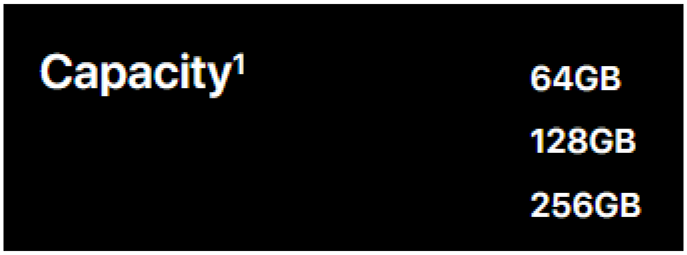
191. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

COUNT IX: PATENT INFRINGEMENT OF THE ’986 PATENT

192. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

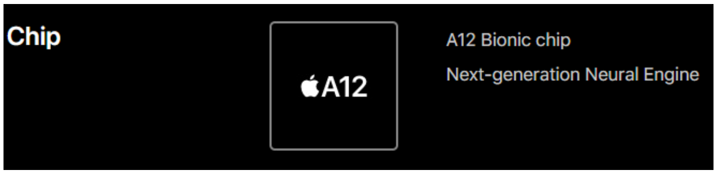
193. Apple infringes (literally and/or under the Doctrine of Equivalents) the ’986 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the ’986 patent including at least the ’986 accused products noted above. In addition, Apple derives revenue from the activities relating to the ’986 accused products.

194. For example and as shown below, the ’986 accused products infringe at least claim 1 of the ’986 patent. For example, per claim 1, to the extent the preamble is limiting, the ’986 accused products satisfy a “first device configured for communicating with a second device associated with the same user, the first device comprising: a memory; a processor configured for:” As shown below by the exemplary evidence, the ’986 accused products, comprise a processor and a memory and a given user’s Apple device (“first device”) is configured to communicate with a user’s other Apple devices (“second device”) using, for example, Apple Handoff and Universal Clipboard.



Capacity[1]	64GB 128GB 256GB

https://www.apple.com/iphone-xr/specs/



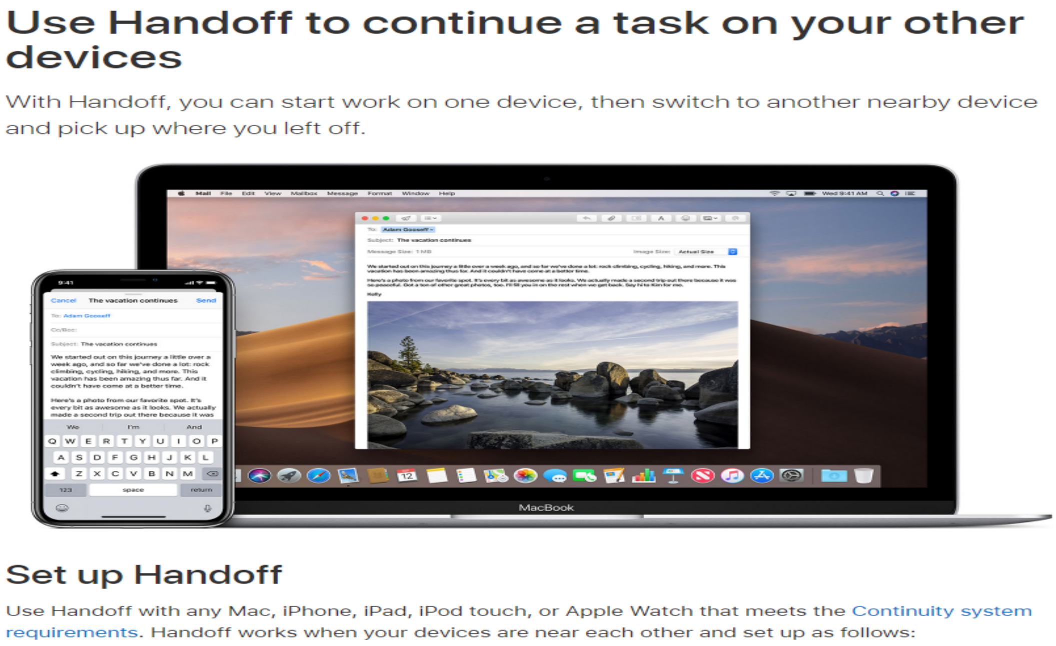
Chip

A12

A12 Bionic chip

Next-generation Neural Engine

https://www.apple.com/iphone-xr/specs/

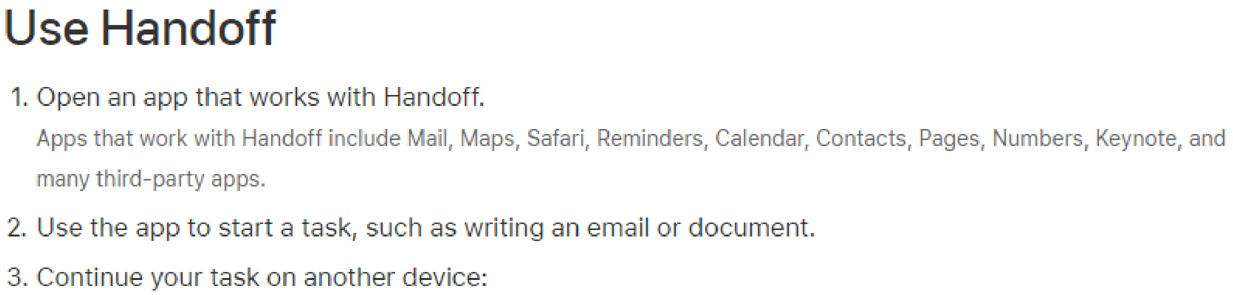


Use Handoff to continue a task on your other devices

With Handoff, you can start work on one device, then switch to another nearby device and pick up where you left off.

Set up Handoff

Use Handoff with any Mac, iPhone, iPad, iPod touch, or Apple Watch that meets the Continuity system requirements. Handoff works when your devices are near each other and set up as follows:



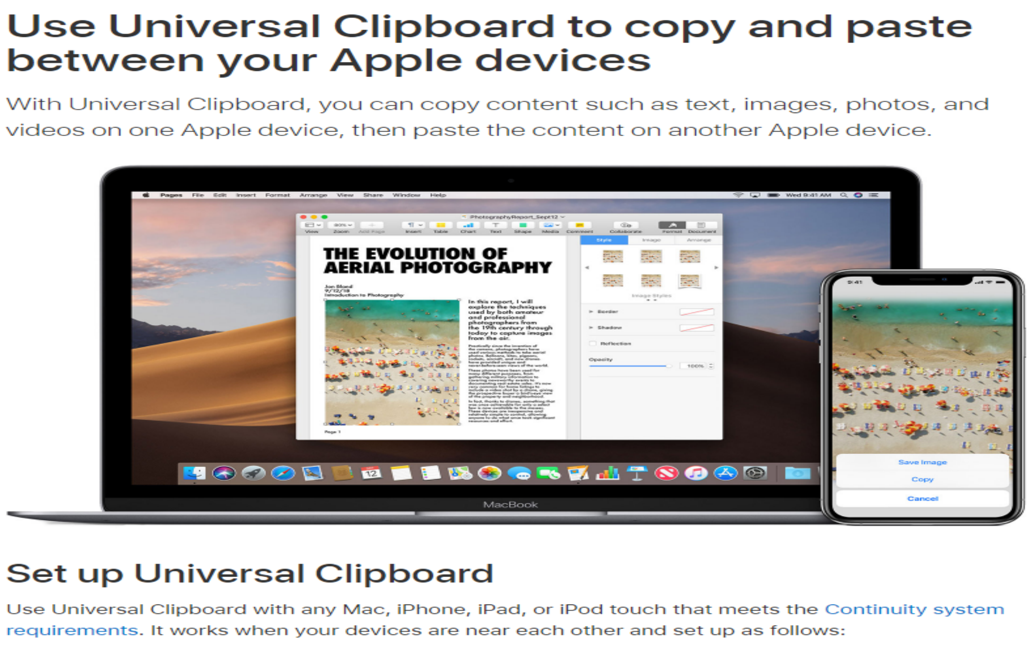
Use Handoff

1. Open an app that works with Handoff.

   Apps that work with Handoff include Mail, Maps, Safari, Reminders, Calendar, Contacts, Pages, Numbers, Keynote, and many third-party apps.

2. Use the app to start a task, such as writing an email or document.
3. Continue your task on another device:

https://support.apple.com/en-us/HT209455

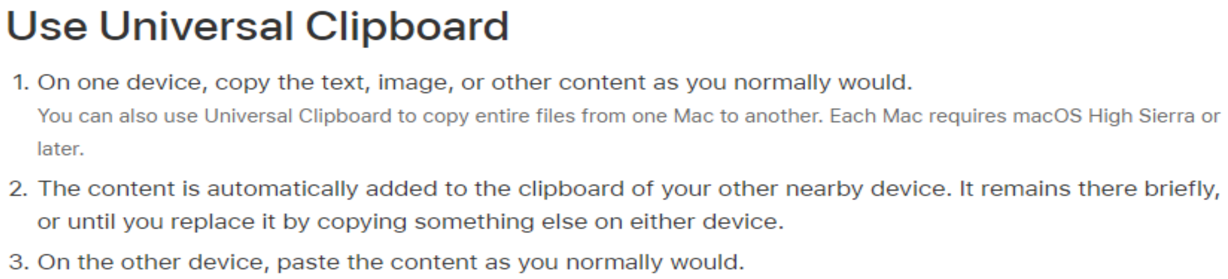


Use Universal Clipboard to copy and paste between your Apple devices

With Universal Clipboard, you can copy content such as text, images, photos, and videos on one Apple device, then paste the content on another Apple device.

Set up Universal Clipboard

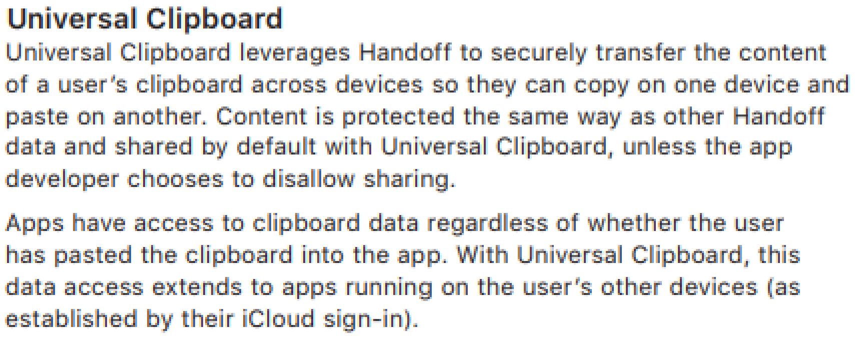
Use Universal Clipboard with any Mac, iPhone, iPad, or iPod touch that meets the Continuity system requirements. It works when your devices are near each other and set up as follows:



Use Universal Clipboard

1. On one device, copy the text, image, or other content as you normally would.
   You can also use Universal Clipboard to copy entire files from one Mac to another. Each Mac requires macOS High Sierra or later.
2. The content is automatically added to the clipboard of your other nearby device. It remains there briefly, or until you replace it by copying something else on either device.
3. On the other device, paste the content as you normally would.

https://support.apple.com/en-us/HT209460



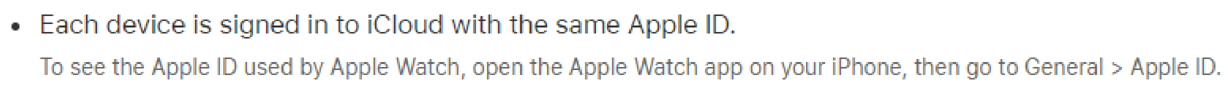
Universal Clipboard

Universal Clipboard leverages Handoff to securely transfer the content of a user's clipboard across devices so they can copy on one device and paste on another. Content is protected the same way as other Handoff data and shared by default with Universal Clipboard, unless the app developer chooses to disallow sharing.

Apps have access to clipboard data regardless of whether the user has pasted the clipboard into the app. With Universal Clipboard, this data access extends to apps running on the user's other devices (as established by their iCloud sign-in).

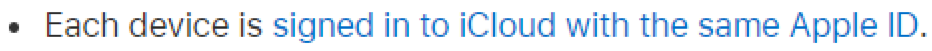
https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

195. Claim 1 of the '986 patent recites "authenticating the first device of a user." The '986 accused products satisfy this limitation. For example, the '986 accused products with Apple's Handoff and Universal Clipboard features require a device to be authenticated using Apple ID, as shown by the exemplary documentation below.



- Each device is signed in to iCloud with the same Apple ID.
  To see the Apple ID used by Apple Watch, open the Apple Watch app on your iPhone, then go to General > Apple ID.

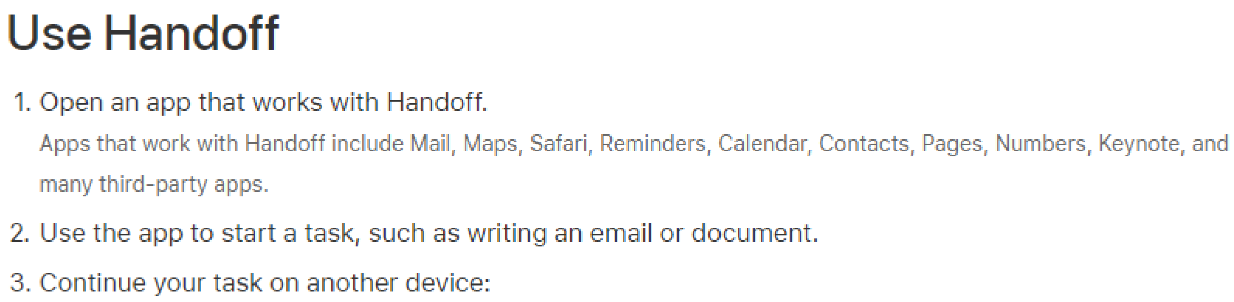
https://support.apple.com/en-us/HT209455



- Each device is signed in to iCloud with the same Apple ID.

https://support.apple.com/en-us/HT209460

196. Claim 1 of the '986 patent recites "accessing content in response to a selection at an application at the first device." The '986 accused products satisfy this limitation. For example, a user accesses content (e.g., Mail, Maps, Safari, Reminders, Calendar, Contacts, Pages, Numbers, or Keynote apps) on the first device, as shown by the exemplary documentation below.

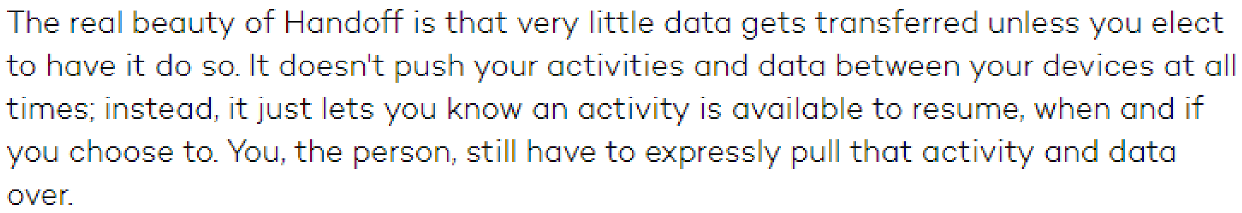


Use Handoff

1. Open an app that works with Handoff.
   Apps that work with Handoff include Mail, Maps, Safari, Reminders, Calendar, Contacts, Pages, Numbers, Keynote, and many third-party apps.
2. Use the app to start a task, such as writing an email or document.
3. Continue your task on another device:

https://support.apple.com/en-us/HT209455

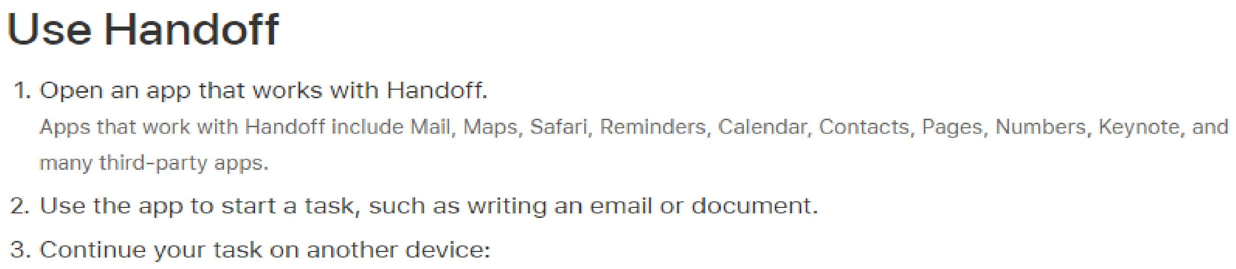
197. Claim 1 of the '986 patent recites "transmitting a representation of the accessed content to an application at a second device associated with the user, wherein the second device is authenticated over a mobile network." The '986 accused products satisfy this limitation. For example, the '986 accused products with Apple's Handoff and Universal Clipboard features transmit a representation of the content to an application at a second device. The second device must be authenticated using the same Apple ID as the first device, as shown by the exemplary documentation below.



The real beauty of Handoff is that very little data gets transferred unless you elect to have it do so. It doesn't push your activities and data between your devices at all times; instead, it just lets you know an activity is available to resume, when and if you choose to. You, the person, still have to expressly pull that activity and data over.

https://www.imore.com/how-apple-keeps-your-handoff-data-secure

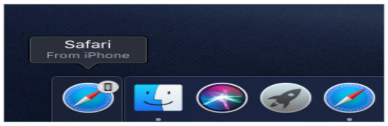
198. Claim 1 of the ’986 patent recites “wherein the content is transmitted to the second device in response to a selection at the second device, wherein the selection identifies accessed content at the first device to be transmitted.” The ’986 accused products satisfy this limitation. For example, Apple’s Handoff and Universal Clipboard features transmit the content to the second device in response to a user selection at the second Apple device, as shown by the exemplary documentation below.



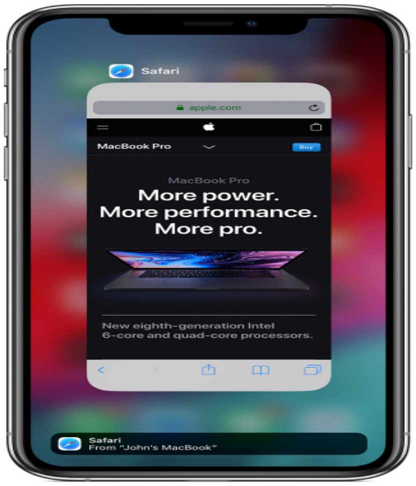
Use Handoff

1. Open an app that works with Handoff.
   Apps that work with Handoff include Mail, Maps, Safari, Reminders, Calendar, Contacts, Pages, Numbers, Keynote, and many third-party apps.
2. Use the app to start a task, such as writing an email or document.
3. Continue your task on another device:

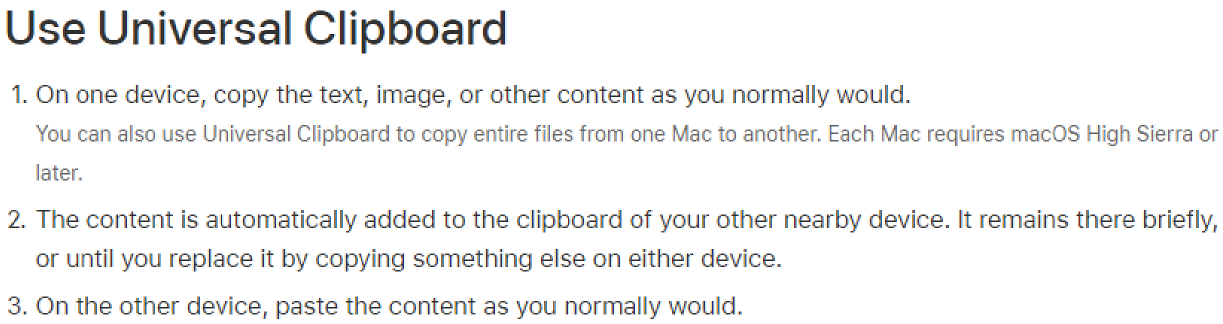
- If you're switching to your Mac, click the app's icon in the Dock:



- If you're switching to your iPhone, iPad, or iPod touch, open the App Switcher, as you would when switching between apps, then tap the app banner at the bottom of the screen.



https://support.apple.com/en-us/HT209455

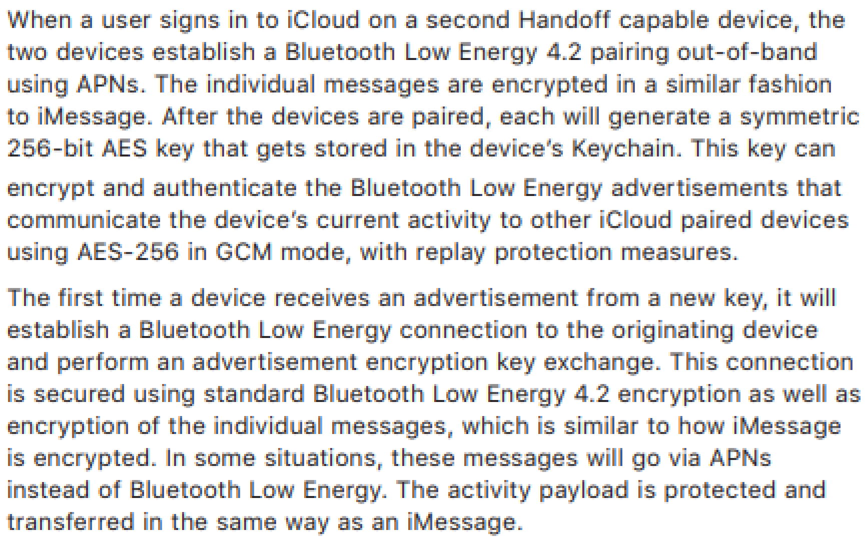


Use Universal Clipboard

1. On one device, copy the text, image, or other content as you normally would.
   You can also use Universal Clipboard to copy entire files from one Mac to another. Each Mac requires macOS High Sierra or later.
2. The content is automatically added to the clipboard of your other nearby device. It remains there briefly, or until you replace it by copying something else on either device.
3. On the other device, paste the content as you normally would.

https://support.apple.com/en-us/HT209460

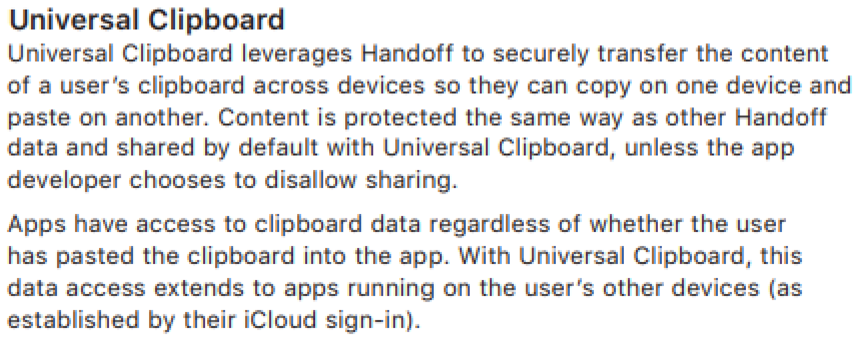
199. Claim 1 of the ’986 patent recites “wherein subsequent to the second device authenticating over the mobile network, the second device is authenticated with the first device before the content is transmitted.” The ’986 accused products satisfy this limitation. For example, the second Apple device must be authenticated with the first Apple device before the content is transmitted, as shown by the exemplary documentation below.



When a user signs in to iCloud on a second Handoff capable device, the two devices establish a Bluetooth Low Energy 4.2 pairing out-of-band using APNs. The individual messages are encrypted in a similar fashion to iMessage. After the devices are paired, each will generate a symmetric 256-bit AES key that gets stored in the device's Keychain. This key can encrypt and authenticate the Bluetooth Low Energy advertisements that communicate the device's current activity to other iCloud paired devices using AES-256 in GCM mode, with replay protection measures.

The first time a device receives an advertisement from a new key, it will establish a Bluetooth Low Energy connection to the originating device and perform an advertisement encryption key exchange. This connection is secured using standard Bluetooth Low Energy 4.2 encryption as well as encryption of the individual messages, which is similar to how iMessage is encrypted. In some situations, these messages will go via APNs instead of Bluetooth Low Energy. The activity payload is protected and transferred in the same way as an iMessage.

***



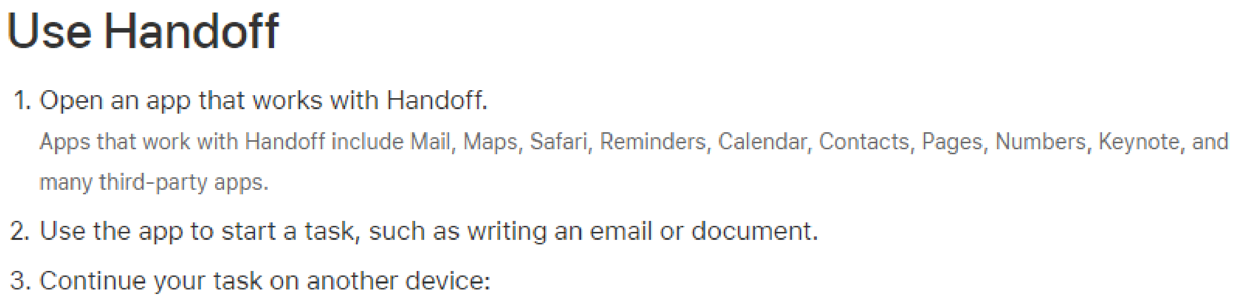
Universal Clipboard

Universal Clipboard leverages Handoff to securely transfer the content of a user's clipboard across devices so they can copy on one device and paste on another. Content is protected the same way as other Handoff data and shared by default with Universal Clipboard, unless the app developer chooses to disallow sharing.

Apps have access to clipboard data regardless of whether the user has pasted the clipboard into the app. With Universal Clipboard, this data access extends to apps running on the user's other devices (as established by their iCloud sign-in).

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

200. Claim 1 of the '986 patent recites "wherein the application at the first device and the application at the second device are branded by a same entity." The '986 accused products satisfy this limitation. For example, the '986 accused products with Apple's Handoff and Universal Clipboard features are used with Apple-branded apps (e.g., Mail, Maps, Safari, Reminders, Calendar, Contacts, Pages, Numbers, Keynote).

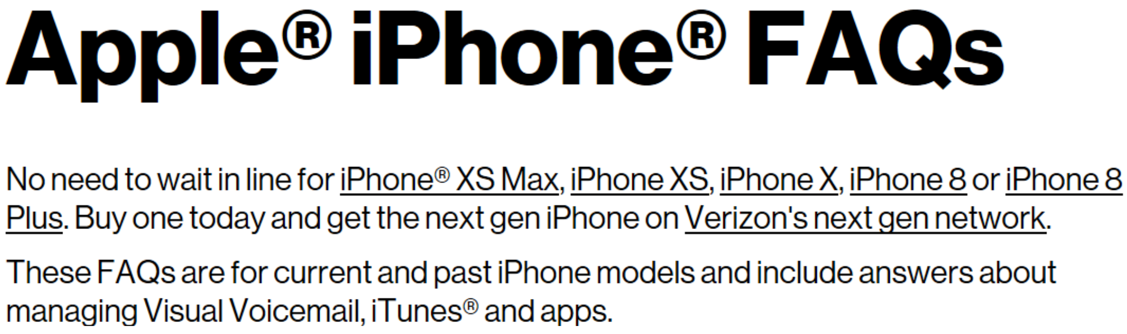


Use Handoff

1. Open an app that works with Handoff.
   Apps that work with Handoff include Mail, Maps, Safari, Reminders, Calendar, Contacts, Pages, Numbers, Keynote, and many third-party apps.
2. Use the app to start a task, such as writing an email or document.
3. Continue your task on another device:

https://support.apple.com/en-us/HT209455

201. Claim 1 of the ’986 patent recites “wherein the same entity is other than a provider that operates the mobile network.” The ’986 accused products satisfy this limitation. For example, the mobile cellular network (e.g., AT&T, Verizon, etc.) is not Apple-branded, as shown by the exemplary documentation below.

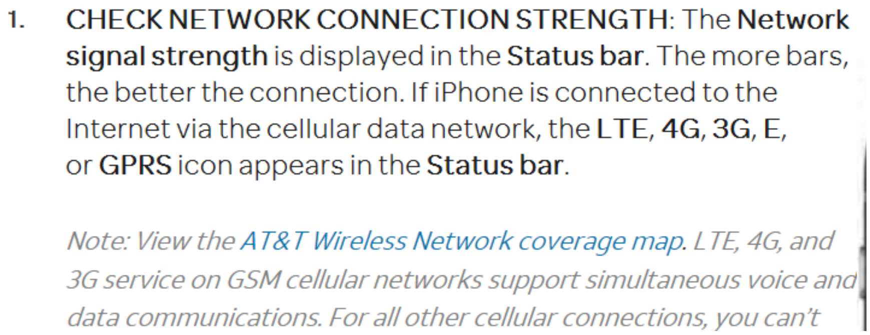


Apple® iPhone® FAQs

No need to wait in line for iPhone® XS Max, iPhone XS, iPhone X, iPhone 8 or iPhone 8 Plus. Buy one today and get the next gen iPhone on Verizon's next gen network.

These FAQs are for current and past iPhone models and include answers about managing Visual Voicemail, iTunes® and apps.

https://www.verizonwireless.com/support/iphone-faqs/



1. **CHECK NETWORK CONNECTION STRENGTH**: The **Network signal strength** is displayed in the **Status bar**. The more bars, the better the connection. If iPhone is connected to the Internet via the cellular data network, the **LTE**, **4G**, **3G**, **E**, or **GPRS** icon appears in the **Status bar**.

   Note: View the AT&T Wireless Network coverage map. LTE, 4G, and 3G service on GSM cellular networks support simultaneous voice and data communications. For all other cellular connections, you can’t

https://www.att.com/devicehowto/tutorial.html#!/stepbystep/id/stepbystep_KM1273413?make=Apple&model=iPhoneXR&gsi=3xf9xt

202. Thus, as illustrated above, the ’986 accused products directly infringe one or more claims of the ’986 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the ’986 patent.

203. Apple indirectly infringes the ’986 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple’s customers and end-

users, in this district and elsewhere in the United States. For example, Apple's customers and end-users directly infringe through their use of the inventions claimed in the '986 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the '986 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the '986 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products' capability in an infringing manner, including through the use Apple's Handoff and Universal Clipboard features. *See, e.g.,* https://support.apple.com/en-us/HT209455; https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf; https://support.apple.com/en-us/HT209460. As a result of Apple's inducement, Apple's customers and end-users use the Apple products in the way Apple intends and directly infringe the '986 patent. Apple performs these affirmative acts with knowledge of the '986 patent and with the intent, or willful blindness, that the induced acts directly infringe the '986 patent. Apple has had knowledge and notice of the '986 patent at least as of the filing of this complaint.

204. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

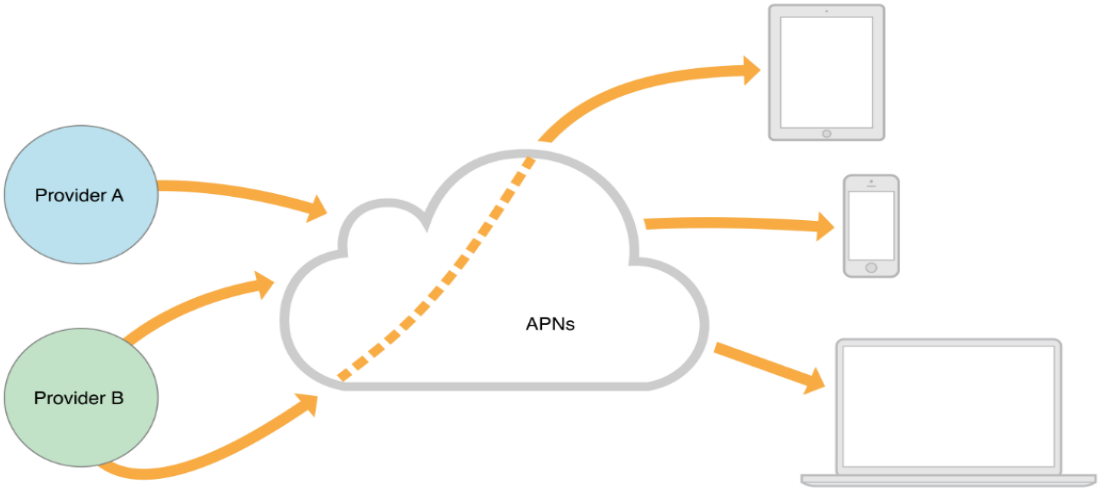
COUNT X: PATENT INFRINGEMENT OF THE '176 PATENT

205. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

206. Apple infringes (literally and/or under the Doctrine of Equivalents) the '176 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the '176 patent including at least the '176 accused products noted above. In addition, Apple derives revenue from the activities relating to the '176 accused products.

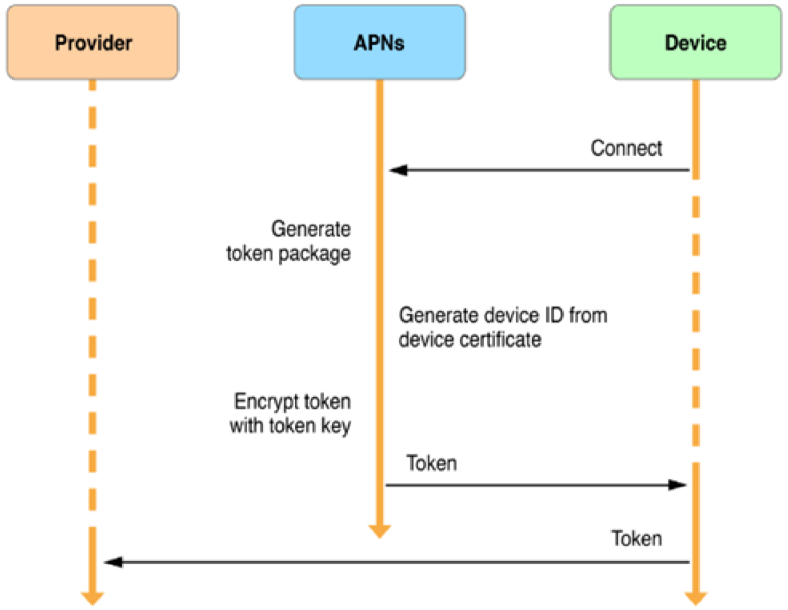
207. For example and as shown below, the '176 accused products infringe at least claim 1 of the '176 patent. To the extent the preamble is limiting, the '176 accused products satisfy a "server for providing access to one or more data stores, comprising: a memory and a processor, the server communicatively coupled to a network and the one or more data stores, wherein the server is configured to:" As shown below by the exemplary evidence, the '176 accused products include an Apple server that provides access to one or more data stores, for example, data stores relating to app providers.

Figure 6-2 Pushing remote notifications from multiple providers to multiple devices



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html

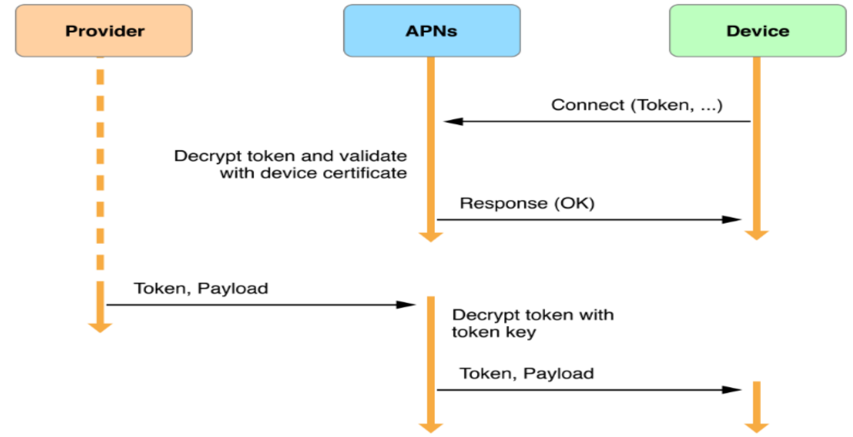
Figure 6-6 Managing the device token



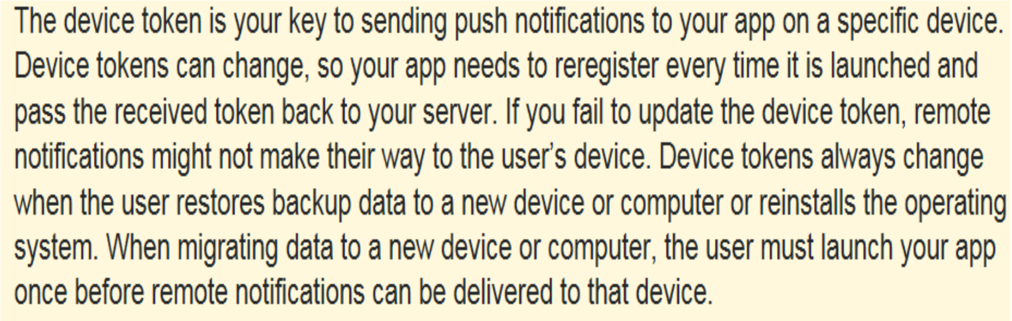
https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

208. Claim 1 of the ’176 patent recites “send a first identifier to a client device upon the client device communicating with the server for the client device to present the first identifier in a subsequent connection with the server.” The ’176 accused products include this feature. For example, Apple’s server sends a first identifier for a client device upon the client device communicating with the server for the client device to present the first identifier in a subsequent connection with the server, as shown in the exemplary documentation and iPhone logs below.

Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1



The device token is your key to sending push notifications to your app on a specific device. Device tokens can change, so your app needs to reregister every time it is launched and pass the received token back to your server. If you fail to update the device token, remote notifications might not make their way to the user's device. Device tokens always change when the user restores backup data to a new device or computer or reinstalls the operating system. When migrating data to a new device or computer, the user must launch your app once before remote notifications can be delivered to that device.

https://stackoverflow.com/questions/6652242/does-the-apns-device-token-ever-change-once-created

```
default	12:32:08.827534 -0400	apsd	2019-03-25 12:32:08 -0400 apsd[85]:
<APSCourier: 0x11bd0d530>: Connected to courier 36-courier.push.apple.com (17.249.76.5)
connection: <APSCourierConnection: 0x11bd21f70> usingPackedFormat YES
secureHandshakeEnabled YES onInterface: NonCellular
default	12:32:08.827619 -0400	apsd	2019-03-25 12:32:08 -0400 apsd[85]:
copyTokenForDomain push.apple.com (null)
default	12:32:08.832767 -0400	apsd	2019-03-25 12:32:08 -0400 apsd[85]:
copyTokenForDomain push.apple.com (null)
default	12:32:08.838639 -0400	apsd	2019-03-25 12:32:08 -0400 apsd[85]:
copyTokenForDomain push.apple.com (null)
default	12:32:08.839255 -0400	apsd	2019-03-25 12:32:08 -0400 apsd[85]:
<APSCourier: 0x11bd0d530> Sending presence message for deviceToken with token YES on
interface: NonCellular tokenName: systemToken
```

iPhone log, captured March 25, 2019

```
default	12:32:09.008560 -0400	apsd	2019-03-25 12:32:08 -0400 apsd[85]:
<APSCourier: 0x11bd0d530>: Sending filter message for: token type systemToken, token:
```

```
<6dc220e2 a40b027a d01c77ce 7df3f958 fdbb9ec5 bf218587 fc444b14 e2e9c4b1>, enabled
hashes {
   <2038739c 2e580f36 13e5e2ed cdb6511f 610988d0> = "com.apple.icloud-
container.com.apple.willowd";
   <61b1ed7e 04116ffc a2b48ccf fa9111a9 f4cb772a> = "com.tencent.xin.voip";
   <e2225610 d1b716ac 87094ad0 2abd95cb 727975fb> = "com.apple.icloud-
container.com.apple.cloudpaird";
   <f690f1b3 413e1123 a4e624b5 36e6ddba 55d6b567> = "com.google.GVDialer.voip";
   <3218784d 5fe3f32e 93cfaa9d 8cf6f6cf 39e6d522> = "com.vzw.pushnotification";
   <ed7660ad 372f623f be99baa4 9ea5b97f d98195b2> = "com.spothero.spothero.voip";
   <94a9e606 83653fdd e5192867 3963d5a0 fca0e8c9> = "ph.telegra.Telegraph.voip";
   <efd8f204 92ab0112 9eef9fb0 a834cecf 23d8143c> = "com.apple.icloud-
container.com.apple.accessibility.heard";
   <7496b146 4c8a18ec b9c487fa 8923aa36 8a0bde4f> = "com.mastercard.tokenizati<…>
```

iPhone log, captured March 25, 2019

```
default 16:23:20.453553 -0400 apsd 2019-03-25 16:23:19 -0400 apsd[87]:
<APSCourier: 0x10021a020>: Stream processing: complete yes, invalid no, length parsed 60,
parameters {
   APSProtocolAppTokenGenerateResponse = 0;
   APSProtocolAppTokenGenerateResponseToken = <f8474454 9fcb9c9f ed0185d9 375577f1
a84291b4 c2f47759 6753f645 5f6744fa>;
   APSProtocolAppTokenGenerateResponseTopicHash = <3218784d 5fe3f32e 93cfaa9d
8cf6f6cf 39e6d522>;
   APSProtocolCommand = 18;
```

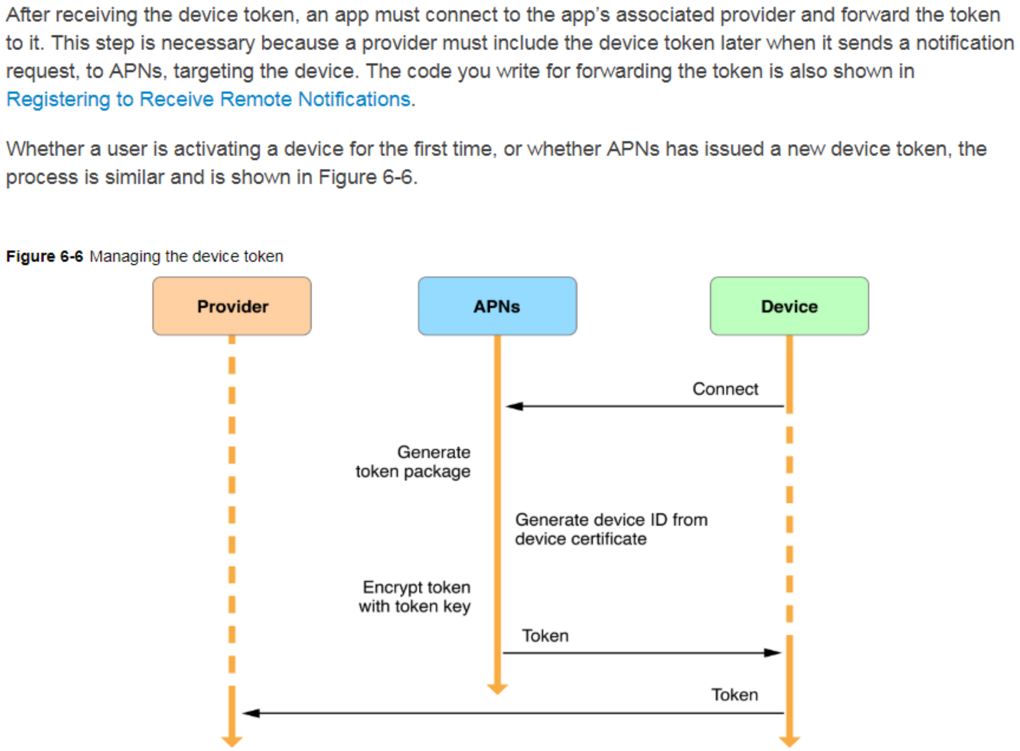
iPhone log, captured March 25, 2019

```
default 12:32:09.008560 -0400 apsd 2019-03-25 12:32:08 -0400 apsd[85]:
<APSCourier: 0x11bd0d530>: Sending filter message for: token type systemToken, token:
<6dc220e2 a40b027a d01c77ce 7df3f958 fdbb9ec5 bf218587 fc444b14 e2e9c4b1>, enabled
hashes {
   <2038739c 2e580f36 13e5e2ed cdb6511f 610988d0> = "com.apple.icloud-
container.com.apple.willowd";
   <61b1ed7e 04116ffc a2b48ccf fa9111a9 f4cb772a> = "com.tencent.xin.voip";
   <e2225610 d1b716ac 87094ad0 2abd95cb 727975fb> = "com.apple.icloud-
container.com.apple.cloudpaird";
   <f690f1b3 413e1123 a4e624b5 36e6ddba 55d6b567> = "com.google.GVDialer.voip";
   <3218784d 5fe3f32e 93cfaa9d 8cf6f6cf 39e6d522> = "com.vzw.pushnotification";
```

iPhone log, captured March 25, 2019

209. Claim 1 of the ’176 patent recites “receive registration information for a data store from the client device, wherein a second identifier is generated and associated with the data store

and the registration information, wherein the second identifier is send to the client device." The '176 accused products include this feature. The '176 accused products receive registration information for a data store from the client device, wherein a second identifier is generated and associated with the data store and the registration information, wherein the second identifier is sent to the client device, as shown in the exemplary documentation below.

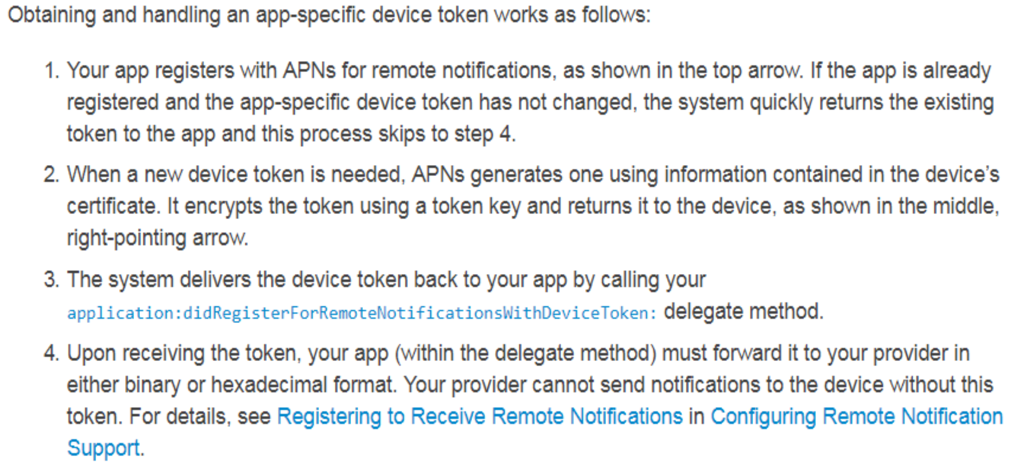


After receiving the device token, an app must connect to the app's associated provider and forward the token to it. This step is necessary because a provider must include the device token later when it sends a notification request, to APNs, targeting the device. The code you write for forwarding the token is also shown in Registering to Receive Remote Notifications.

Whether a user is activating a device for the first time, or whether APNs has issued a new device token, the process is similar and is shown in Figure 6-6.

Figure 6-6 Managing the device token

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1



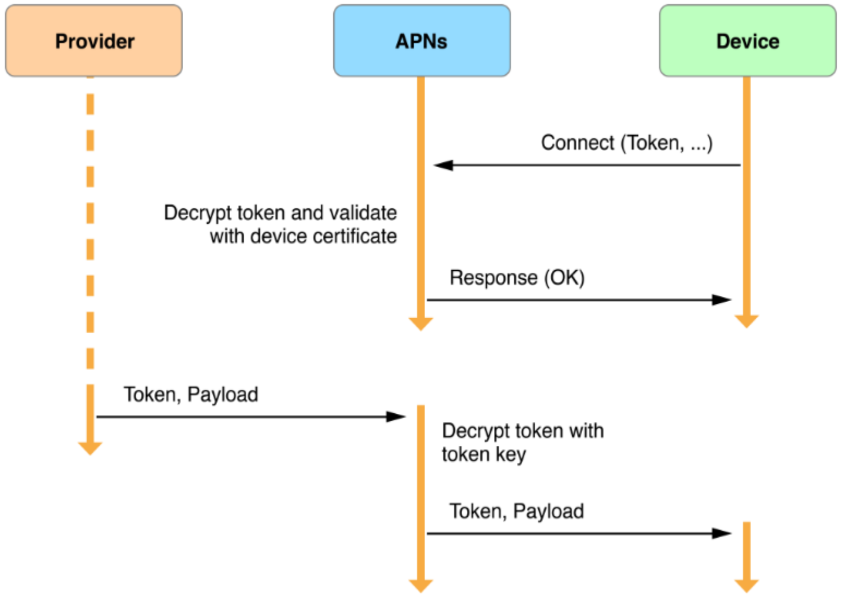
Obtaining and handling an app-specific device token works as follows:

1. Your app registers with APNs for remote notifications, as shown in the top arrow. If the app is already registered and the app-specific device token has not changed, the system quickly returns the existing token to the app and this process skips to step 4.
2. When a new device token is needed, APNs generates one using information contained in the device's certificate. It encrypts the token using a token key and returns it to the device, as shown in the middle, right-pointing arrow.
3. The system delivers the device token back to your app by calling your `application:didRegisterForRemoteNotificationsWithDeviceToken:` delegate method.
4. Upon receiving the token, your app (within the delegate method) must forward it to your provider in either binary or hexadecimal format. Your provider cannot send notifications to the device without this token. For details, see Registering to Receive Remote Notifications in Configuring Remote Notification Support.

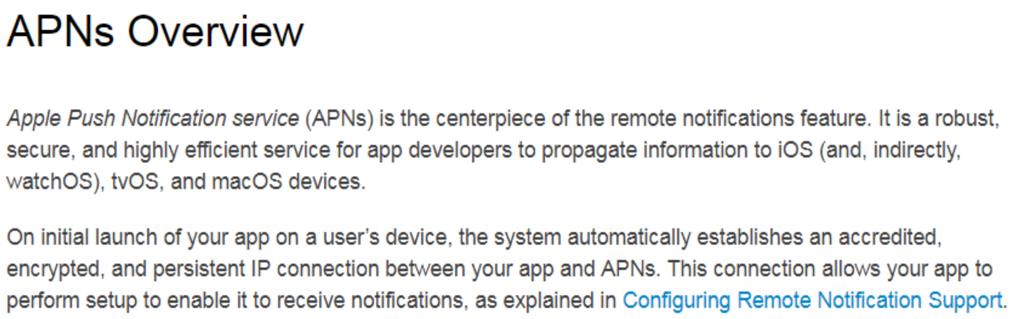
https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

210. Claim 1 of the ’176 patent recites “receive, via the subsequent connection with the client device, a request for the client device to receive information from the data store, wherein the subsequent connection includes the first identifier.” The ’176 accused products satisfy this limitation. The ’176 accused products receive a subsequent connection to receive data from the data store, wherein the subsequent connection includes the first identifier, as shown in the exemplary documentation below, as well as the iPhone logs cited above for the “send a first identifier…” limitation.

Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1



APNs Overview

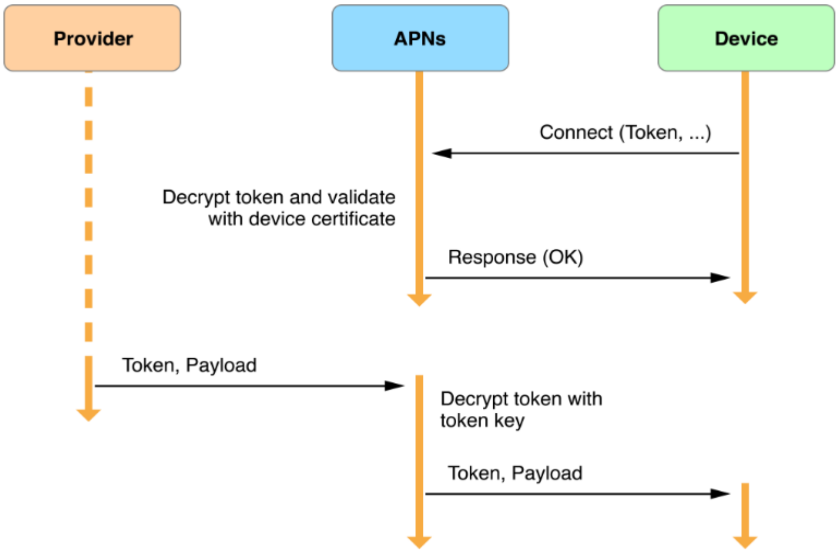
Apple Push Notification service (APNs) is the centerpiece of the remote notifications feature. It is a robust, secure, and highly efficient service for app developers to propagate information to iOS (and, indirectly, watchOS), tvOS, and macOS devices.

On initial launch of your app on a user’s device, the system automatically establishes an accredited, encrypted, and persistent IP connection between your app and APNs. This connection allows your app to perform setup to enable it to receive notifications, as explained in Configuring Remote Notification Support.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

211. Claim 1 of the '176 patent recites "configure a service to receive data from the data store on behalf of the client device, wherein the service is based on the second identifier." The '176 accused products include this feature. The '176 accused products configure a service to receive data from the data store on behalf of the client device, wherein the service is based on the second identifier. For example, the Apple server configures a service to receive data from the data store on behalf of the client device, wherein the service is based on the second identifier, as shown in the exemplary documentation below.

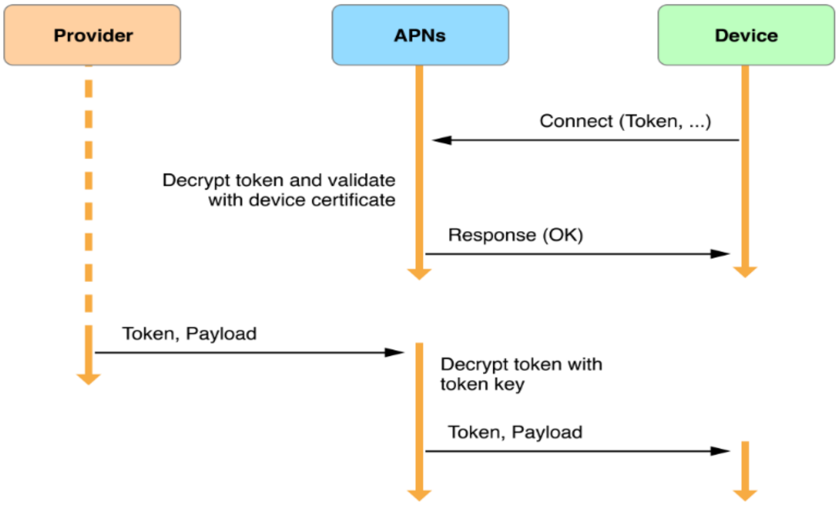
Figure 6-7 Remote notification path from provider to device



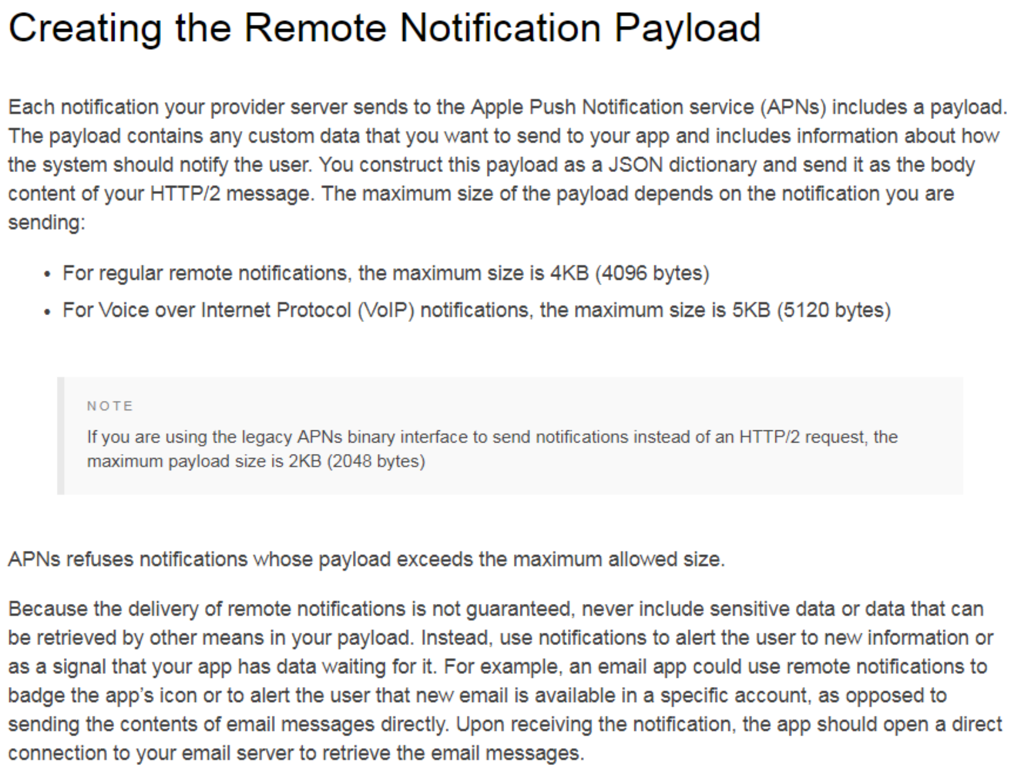
https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

212. Claim 1 of the '176 patent recites "receive a first message indicative of new information at the data store." The '176 accused products include this feature. The '176 accused products receives a first message indicative of new information at the data store, as shown in the exemplary documentation below.

Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1



Creating the Remote Notification Payload

Each notification your provider server sends to the Apple Push Notification service (APNs) includes a payload. The payload contains any custom data that you want to send to your app and includes information about how the system should notify the user. You construct this payload as a JSON dictionary and send it as the body content of your HTTP/2 message. The maximum size of the payload depends on the notification you are sending:

- For regular remote notifications, the maximum size is 4KB (4096 bytes)
- For Voice over Internet Protocol (VoIP) notifications, the maximum size is 5KB (5120 bytes)

> NOTE
>
> If you are using the legacy APNs binary interface to send notifications instead of an HTTP/2 request, the maximum payload size is 2KB (2048 bytes)

APNs refuses notifications whose payload exceeds the maximum allowed size.

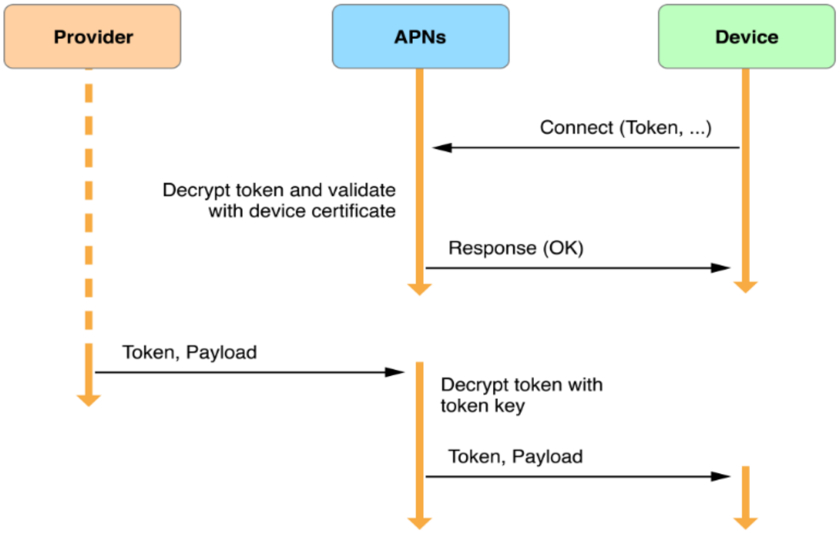
Because the delivery of remote notifications is not guaranteed, never include sensitive data or data that can be retrieved by other means in your payload. Instead, use notifications to alert the user to new information or as a signal that your app has data waiting for it. For example, an email app could use remote notifications to badge the app's icon or to alert the user that new email is available in a specific account, as opposed to sending the contents of email messages directly. Upon receiving the notification, the app should open a direct connection to your email server to retrieve the email messages.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/CreatingtheNotificationPayload.html#//apple_ref/doc/uid/TP40008194-CH10-SW1

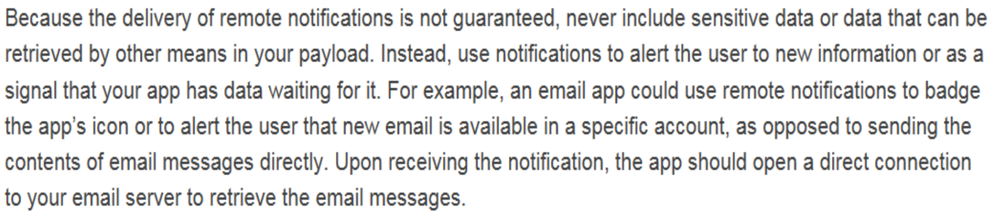
213. Claim 1 of the '176 patent recites "transmit a second message to the client device in response to receipt of the first message." The '176 accused products satisfy this limitation.

The ’176 accused products transmit a second message to the client device in response to receipt of the first message. For example, the Apple server transmits a second message to the client device in response to receipt of the first message.

Figure 6-7 Remote notification path from provider to device

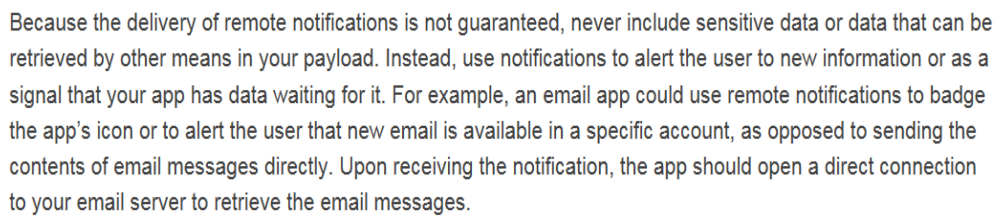


https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1


Because the delivery of remote notifications is not guaranteed, never include sensitive data or data that can be retrieved by other means in your payload. Instead, use notifications to alert the user to new information or as a signal that your app has data waiting for it. For example, an email app could use remote notifications to badge the app’s icon or to alert the user that new email is available in a specific account, as opposed to sending the contents of email messages directly. Upon receiving the notification, the app should open a direct connection to your email server to retrieve the email messages.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/CreatingtheNotificationPayload.html#//apple_ref/doc/uid/TP40008194-CH10-SW1

214. Claim 1 of the ’176 patent recites “wherein additional information associated with the first message is sent from the data store to the client device upon receipt of the second message by the client device.” The ’176 accused products satisfy this limitation. For example, additional information associated with the first message is sent from the data store to the client device upon receipt of the second message by the client device, as shown by the exemplary documentation below.



Because the delivery of remote notifications is not guaranteed, never include sensitive data or data that can be retrieved by other means in your payload. Instead, use notifications to alert the user to new information or as a signal that your app has data waiting for it. For example, an email app could use remote notifications to badge the app's icon or to alert the user that new email is available in a specific account, as opposed to sending the contents of email messages directly. Upon receiving the notification, the app should open a direct connection to your email server to retrieve the email messages.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/CreatingtheNotificationPayload.html#//apple_ref/doc/uid/TP40008194-CH10-SW1

215. Thus, as illustrated above, the ’176 accused products directly infringe one or more claims of the ’176 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the ’176 patent.

216. Apple indirectly infringes the ’176 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple’s customers and end-users, in this district and elsewhere in the United States. For example, Apple’s customers and end-users directly infringe through their use of the inventions claimed in the ’176 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’176 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the ’176 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products’ capability in an infringing manner,

including through the use of Apple's APN service. *See, e.g.*, https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1; https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/CreatingtheNotificationPayload.html#//apple_ref/doc/uid/TP40008194-CH10-SW1; https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf. As a result of Apple's inducement, Apple's customers and end-users use the Apple products in the way Apple intends and directly infringe the '176 patent. Apple performs these affirmative acts with knowledge of the '176 patent and with the intent, or willful blindness, that the induced acts directly infringe the '176 patent. Apple has had knowledge and notice of the '176 patent at least as of the filing of this complaint.

217. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

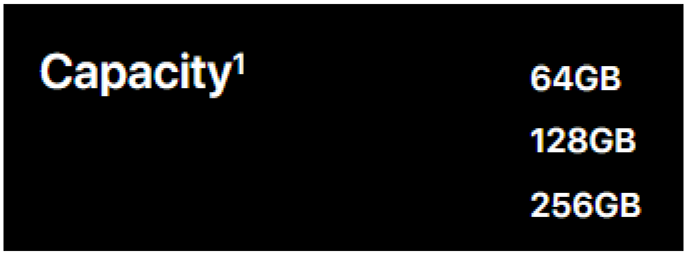
COUNT XI: PATENT INFRINGEMENT OF THE '619 PATENT

218. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

219. Apple infringes (literally and/or under the Doctrine of Equivalents) the '619 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the '619 patent including at least the '619 accused products noted above. In addition, Apple derives revenue from the activities relating to the '619 accused products.

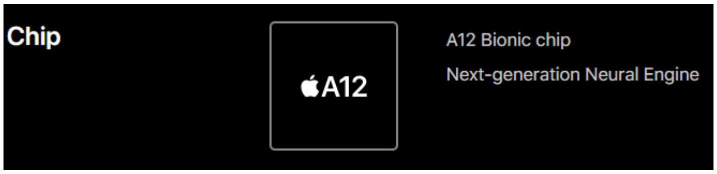
220. For example and as shown below, the '619 accused products infringe at least claim 22 of the '619 patent. For example, per claim 22, to the extent the preamble is limiting,

the ’619 accused products satisfy a “device comprising: a radio; a processor and memory containing instructions executable by the processor whereby the device is operable to:” As shown below by the exemplary evidence, the ’619 accused products, comprise a radio, a processor, and a memory, as shown by the exemplary documentation below. ’619 accused products are configured to exchange information with, for example, Apple Watches.



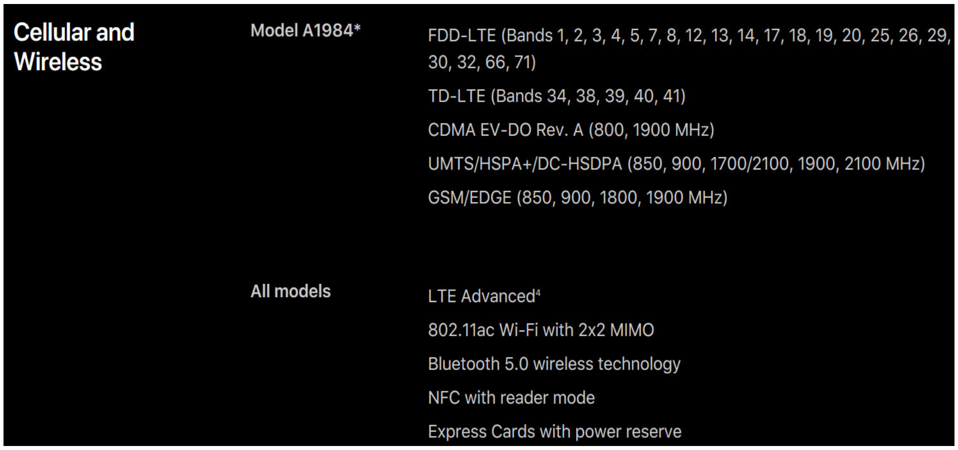
Capacity[1]	64GB 128GB 256GB

https://www.apple.com/iphone-xr/specs/



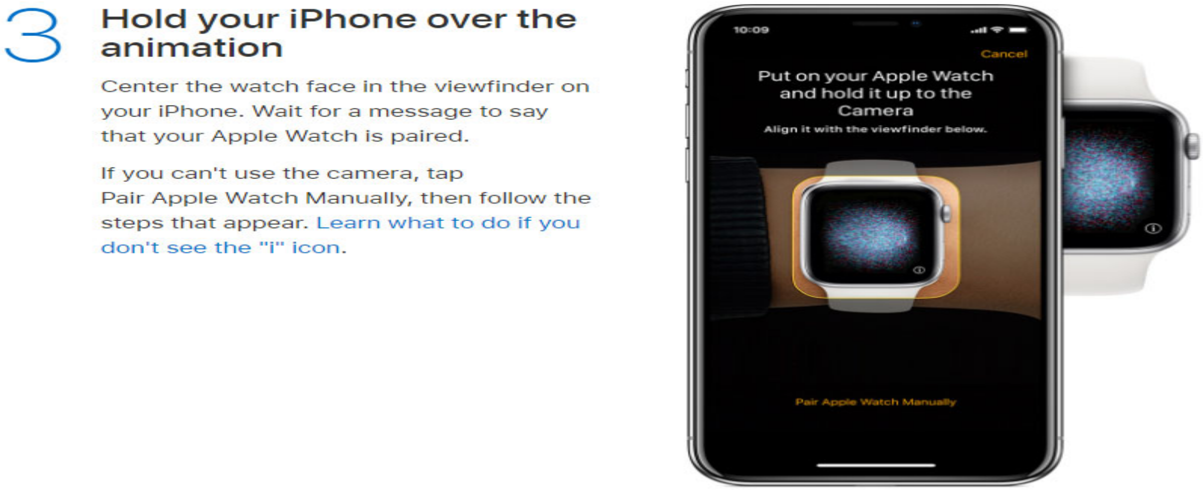
Chip	A12	A12 Bionic chip Next-generation Neural Engine

https://www.apple.com/iphone-xr/specs/



Cellular and Wireless	Model A1984*	FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41) CDMA EV-DO Rev. A (800, 1900 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
	All models	LTE Advanced[4] 802.11ac Wi-Fi with 2x2 MIMO Bluetooth 5.0 wireless technology NFC with reader mode Express Cards with power reserve

https://www.apple.com/iphone-xr/specs/



3 Hold your iPhone over the animation

Center the watch face in the viewfinder on your iPhone. Wait for a message to say that your Apple Watch is paired.

If you can't use the camera, tap Pair Apple Watch Manually, then follow the steps that appear. Learn what to do if you don't see the "i" icon.

Put on your Apple Watch and hold it up to the Camera

Align it with the viewfinder below.

https://support.apple.com/en-us/HT204505

221. Claim 22 of the ’619 patent recites “optically receive information including a displayed service activation code from a remote device.” The ’619 accused products satisfy this limitation. For example, the ’619 accused products optically receives a service activation code from a remote device, such as for example an Apple Watch, as shown by the exemplary documentation below.

3 Hold your iPhone over the animation

Center the watch face in the viewfinder on your iPhone. Wait for a message to say that your Apple Watch is paired.

If you can't use the camera, tap Pair Apple Watch Manually, then follow the steps that appear. Learn what to do if you don't see the "i" icon.

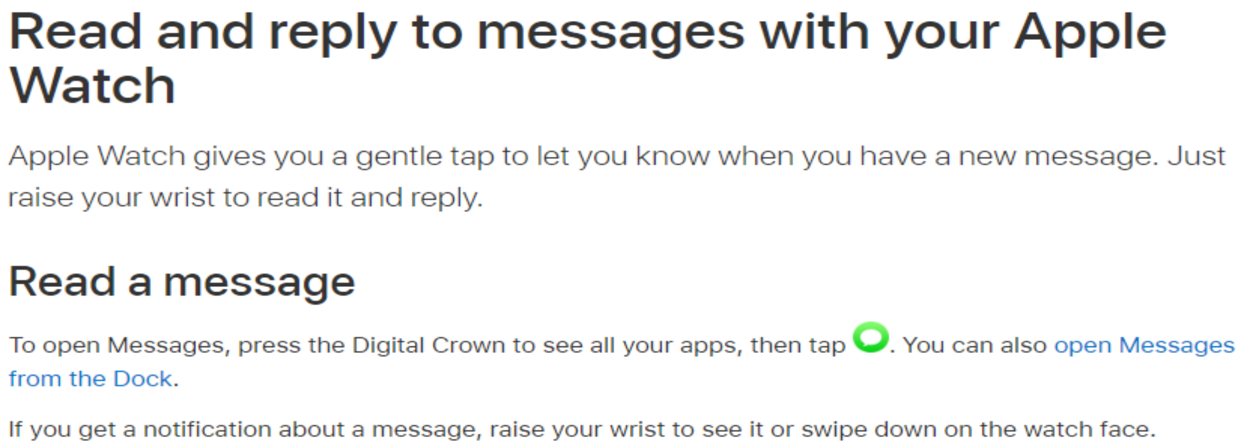
Put on your Apple Watch and hold it up to the Camera

Align it with the viewfinder below.

https://support.apple.com/en-us/HT204505

222. Claim 22 of the ’619 patent recites “register the remote device for access to a messaging account using the service activation code.” The ’619 accused products satisfy this

limitation. The ’619 accused products register, for example an Apple Watch, for access to for example text messages and/or email, as shown by the exemplary documentation below.



Read and reply to messages with your Apple Watch

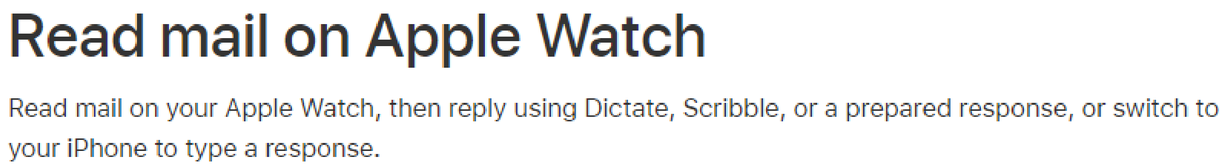
Apple Watch gives you a gentle tap to let you know when you have a new message. Just raise your wrist to read it and reply.

Read a message

To open Messages, press the Digital Crown to see all your apps, then tap 💬. You can also open Messages from the Dock.

If you get a notification about a message, raise your wrist to see it or swipe down on the watch face.

https://support.apple.com/en-us/HT205783

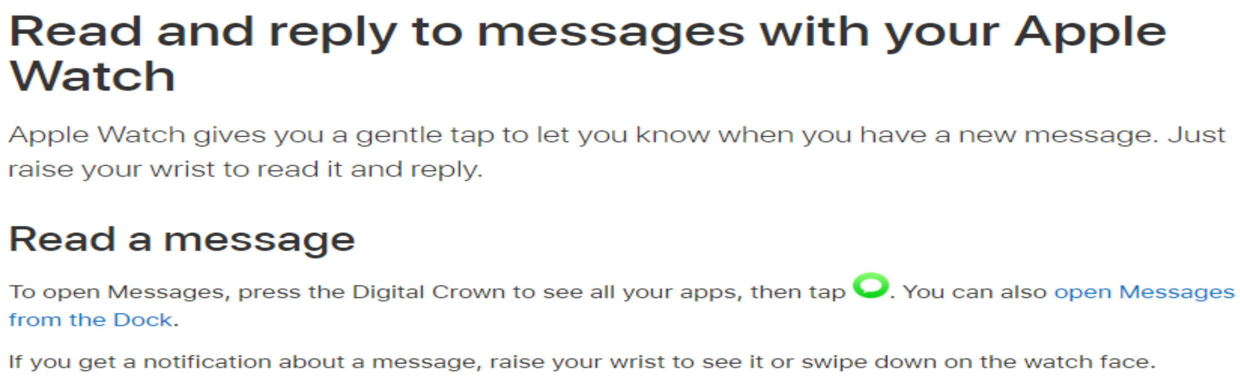


Read mail on Apple Watch

Read mail on your Apple Watch, then reply using Dictate, Scribble, or a prepared response, or switch to your iPhone to type a response.

https://support.apple.com/guide/watch/read-mail-apddca457a4f/watchos

223. Claim 22 of the ’619 patent recites “receive a message for the messaging account.” The ’619 accused products satisfy this limitation. For example, the ’619 accused products receive a message for the messaging account, as shown by the exemplary documentation below.



Read and reply to messages with your Apple Watch

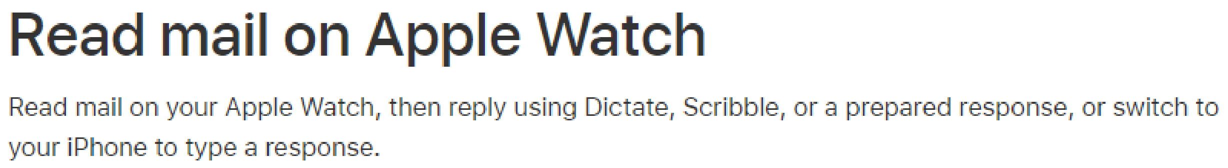
Apple Watch gives you a gentle tap to let you know when you have a new message. Just raise your wrist to read it and reply.

Read a message

To open Messages, press the Digital Crown to see all your apps, then tap 💬. You can also open Messages from the Dock.

If you get a notification about a message, raise your wrist to see it or swipe down on the watch face.

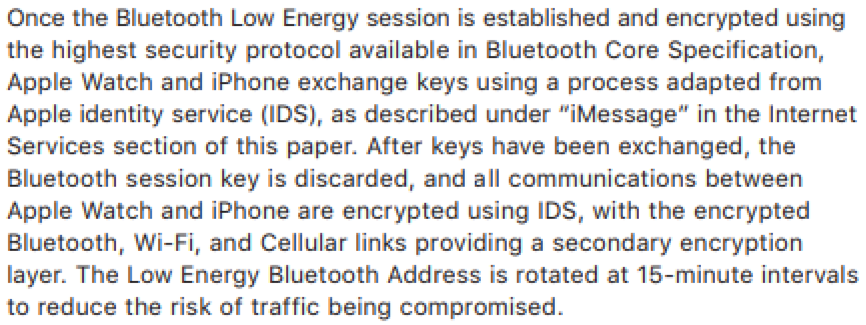
https://support.apple.com/en-us/HT205783



Read mail on Apple Watch

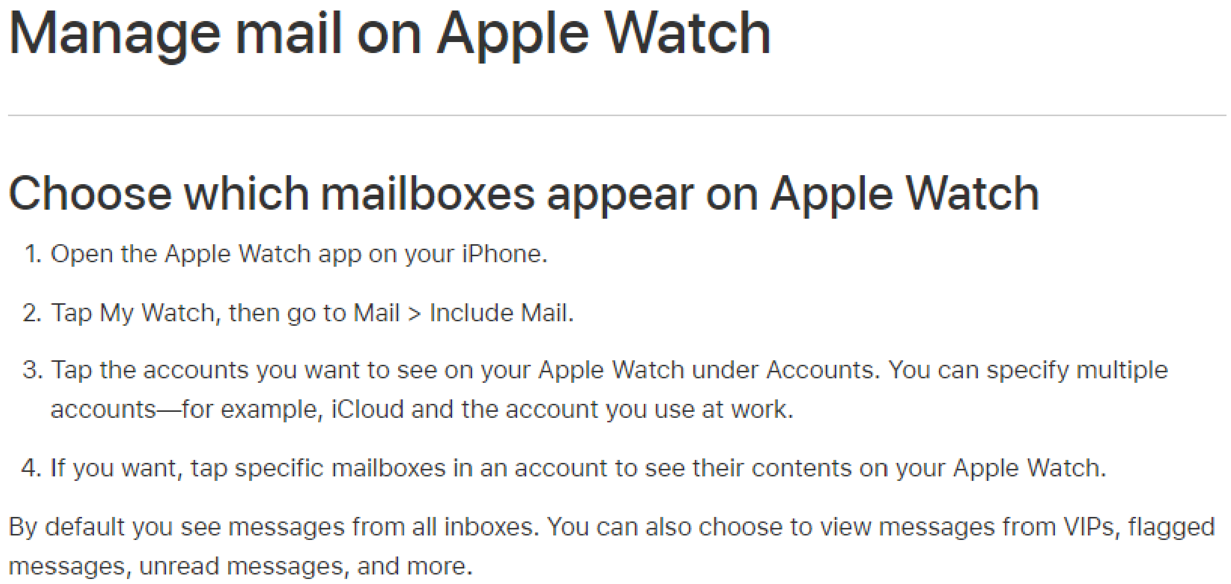
Read mail on your Apple Watch, then reply using Dictate, Scribble, or a prepared response, or switch to your iPhone to type a response.

https://support.apple.com/guide/watch/read-mail-apddca457a4f/watchos



Once the Bluetooth Low Energy session is established and encrypted using the highest security protocol available in Bluetooth Core Specification, Apple Watch and iPhone exchange keys using a process adapted from Apple identity service (IDS), as described under "iMessage" in the Internet Services section of this paper. After keys have been exchanged, the Bluetooth session key is discarded, and all communications between Apple Watch and iPhone are encrypted using IDS, with the encrypted Bluetooth, Wi-Fi, and Cellular links providing a secondary encryption layer. The Low Energy Bluetooth Address is rotated at 15-minute intervals to reduce the risk of traffic being compromised.

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf



Manage mail on Apple Watch

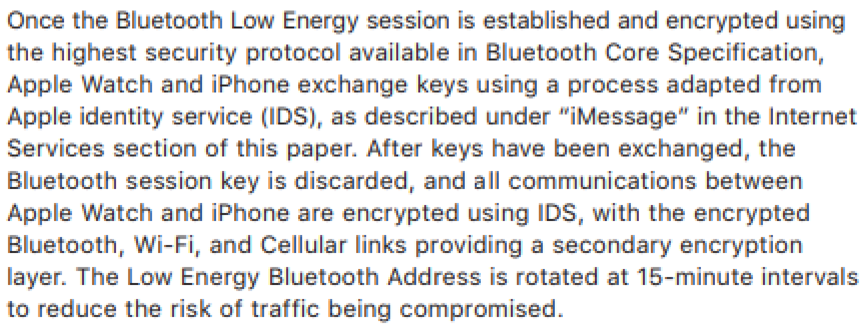
Choose which mailboxes appear on Apple Watch

1. Open the Apple Watch app on your iPhone.
2. Tap My Watch, then go to Mail > Include Mail.
3. Tap the accounts you want to see on your Apple Watch under Accounts. You can specify multiple accounts—for example, iCloud and the account you use at work.
4. If you want, tap specific mailboxes in an account to see their contents on your Apple Watch.

By default you see messages from all inboxes. You can also choose to view messages from VIPs, flagged messages, unread messages, and more.

https://support.apple.com/guide/watch/manage-mail-apd29461a7b7/watchos

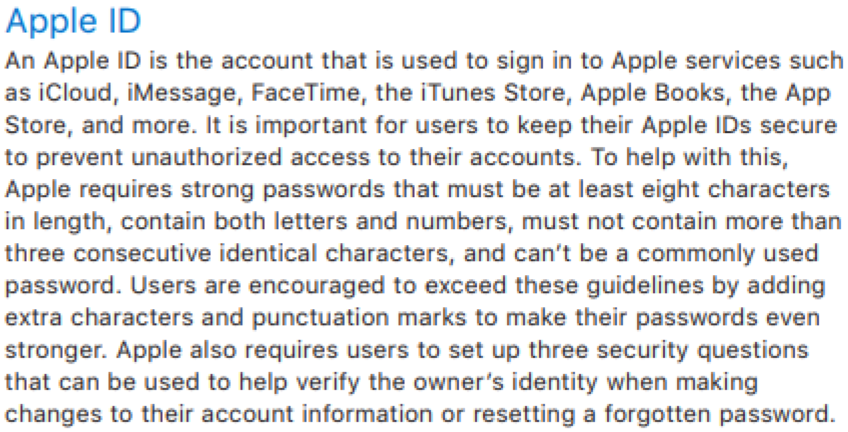
224. Claim 22 of the '619 patent recites "encrypt the message using an encryption key; and send the message to the remote device." The '619 accused products satisfy this limitation. For example, the '619 accused products, encrypt the message and send them to the remote device, such as for example an Apple Watch.



Once the Bluetooth Low Energy session is established and encrypted using the highest security protocol available in Bluetooth Core Specification, Apple Watch and iPhone exchange keys using a process adapted from Apple identity service (IDS), as described under "iMessage" in the Internet Services section of this paper. After keys have been exchanged, the Bluetooth session key is discarded, and all communications between Apple Watch and iPhone are encrypted using IDS, with the encrypted Bluetooth, Wi-Fi, and Cellular links providing a secondary encryption layer. The Low Energy Bluetooth Address is rotated at 15-minute intervals to reduce the risk of traffic being compromised.

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

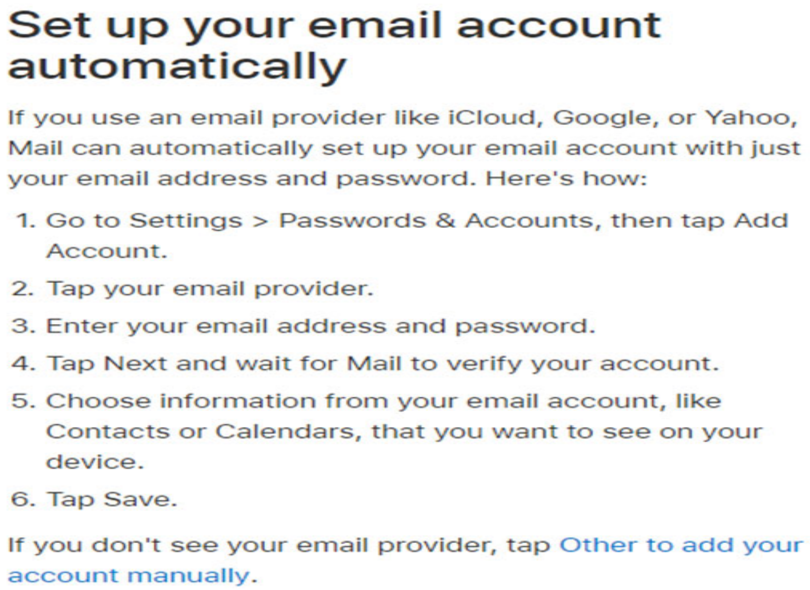
225. Claim 22 of the ’619 patent recites “wherein the device is authenticated to access the messaging account.” The ’619 accused products satisfy this limitation. For example, the device is authenticated to access the messaging account, as shown by the exemplary documentation below.



Apple ID

An Apple ID is the account that is used to sign in to Apple services such as iCloud, iMessage, FaceTime, the iTunes Store, Apple Books, the App Store, and more. It is important for users to keep their Apple IDs secure to prevent unauthorized access to their accounts. To help with this, Apple requires strong passwords that must be at least eight characters in length, contain both letters and numbers, must not contain more than three consecutive identical characters, and can't be a commonly used password. Users are encouraged to exceed these guidelines by adding extra characters and punctuation marks to make their passwords even stronger. Apple also requires users to set up three security questions that can be used to help verify the owner's identity when making changes to their account information or resetting a forgotten password.

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf



Set up your email account automatically

If you use an email provider like iCloud, Google, or Yahoo, Mail can automatically set up your email account with just your email address and password. Here's how:

1. Go to Settings > Passwords & Accounts, then tap Add Account.
2. Tap your email provider.
3. Enter your email address and password.
4. Tap Next and wait for Mail to verify your account.
5. Choose information from your email account, like Contacts or Calendars, that you want to see on your device.
6. Tap Save.

If you don't see your email provider, tap Other to add your account manually.

https://support.apple.com/en-us/HT201320

226. Thus, as illustrated above, the ’619 accused products directly infringe one or more claims of the ’619 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the ’619 patent.

227. Apple indirectly infringes the ’619 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple’s customers and end-users, in this district and elsewhere in the United States. For example, Apple’s customers and end-users directly infringe through their use of the inventions claimed in the ’619 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’619 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the ’619 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World

Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products' capability in an infringing manner, including ncluding through the use of Apple's iPhones with Apple Watches in an infringing manner. *See, e.g.,* https://support.apple.com/en-us/HT205783; https://support.apple.com/guide/watch/read-mail-apddca457a4f/watchos; https://support.apple.com/guide/watch/read-mail-apddca457a4f/watchos. As a result of Apple's inducement, Apple's customers and end-users use the Apple products in the way Apple intends and directly infringe the '619 patent. Apple performs these affirmative acts with knowledge of the '619 patent and with the intent, or willful blindness, that the induced acts directly infringe the '619 patent. Apple has had knowledge and notice of the '619 patent at least as of the filing of this complaint.

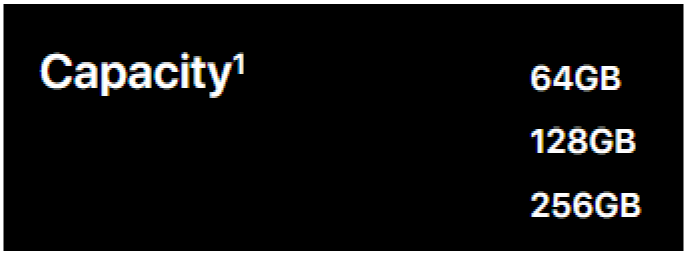
228. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

COUNT XII: PATENT INFRINGEMENT OF THE '029 PATENT

229. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

230. Apple infringes (literally and/or under the Doctrine of Equivalents) the '029 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the '029 patent including at least the '029 accused products noted above. In addition, Apple derives revenue from the activities relating to the '029 accused products.

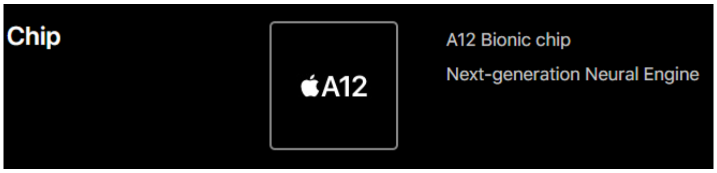
231. For example and as shown below, the ’029 accused products infringe at least claim 1 of the ’029 patent. For example, to the extent the preamble is limiting, the ’029 accused products satisfy a “mobile device having an established multiplexed connection for optimizing communications, the mobile device comprising: a memory; and a processor configured for:” As shown below by the exemplary evidence, the ’029 accused products satisfy a mobile device having an established multiplexed connection for optimizing communications and comprise a memory and a processor. For example, through the Apple Push Notification service (APNs), iOS devices have an established multiplexed connection for optimizing communications, allowing, for example, multiple providers to use a common connection to send push notifications to their corresponding applications on the device, as shown in the exemplary documentation below.



Capacity[1]

64GB
128GB
256GB

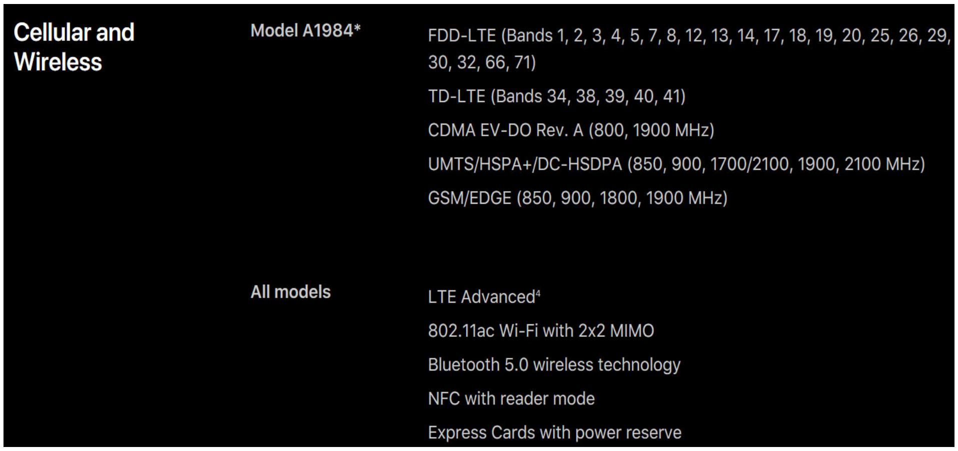
https://www.apple.com/iphone-xr/specs/



Chip

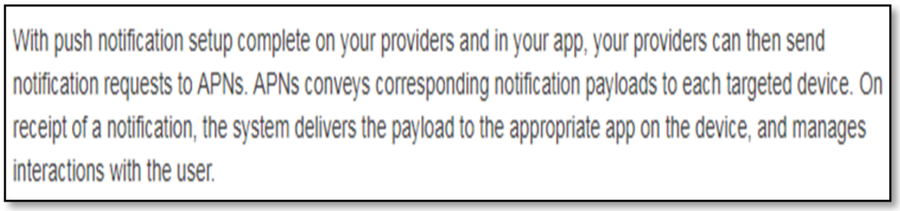
A12 Bionic chip
Next-generation Neural Engine

https://www.apple.com/iphone-xr/specs/



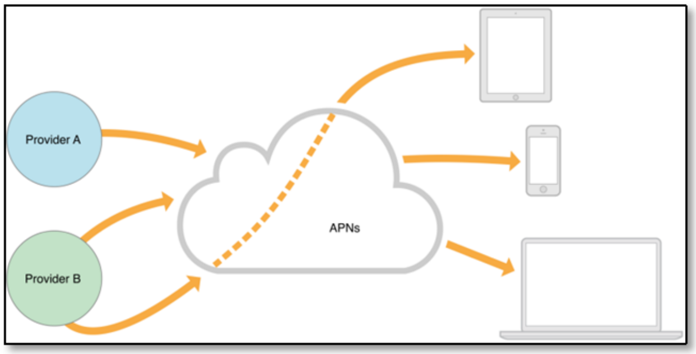
Cellular and Wireless	Model A1984*	FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41) CDMA EV-DO Rev. A (800, 1900 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
	All models	LTE Advanced[4] 802.11ac Wi-Fi with 2x2 MIMO Bluetooth 5.0 wireless technology NFC with reader mode Express Cards with power reserve

https://www.apple.com/iphone-xr/specs/



With push notification setup complete on your providers and in your app, your providers can then send notification requests to APNs. APNs conveys corresponding notification payloads to each targeted device. On receipt of a notification, the system delivers the payload to the appropriate app on the device, and manages interactions with the user.

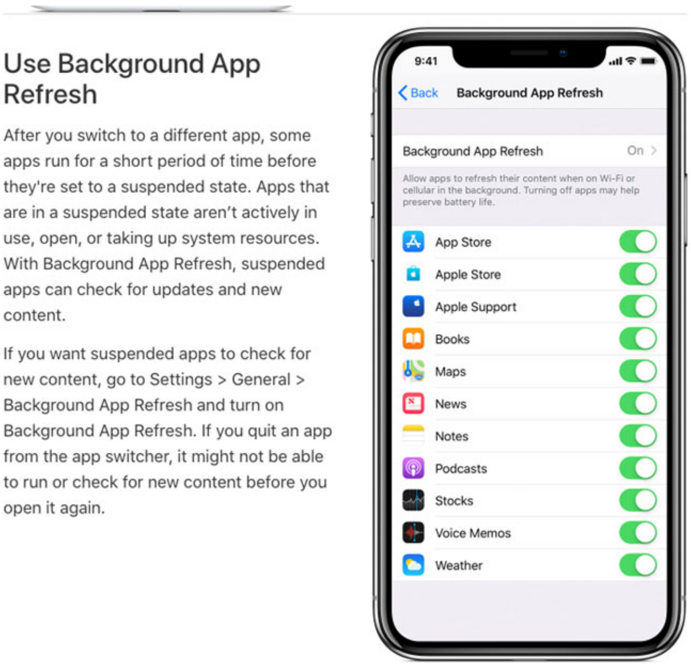
https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html

232. Claim 1 of the ’029 patent recites “receiving a selection from a user whether to enable an application for fetching.” The ’029 accused products satisfy this limitation. For example, the ’029 accused products are configured for receiving a selection from a user whether to enable an application for fetching. For example, the ’029 accused products provide a screen

allowing a user to choose which applications are allowed for background execution, as shown in the exemplary documentation below.



Use Background App Refresh

After you switch to a different app, some apps run for a short period of time before they're set to a suspended state. Apps that are in a suspended state aren't actively in use, open, or taking up system resources. With Background App Refresh, suspended apps can check for updates and new content.

If you want suspended apps to check for new content, go to Settings > General > Background App Refresh and turn on Background App Refresh. If you quit an app from the app switcher, it might not be able to run or check for new content before you open it again.

Background App Refresh On

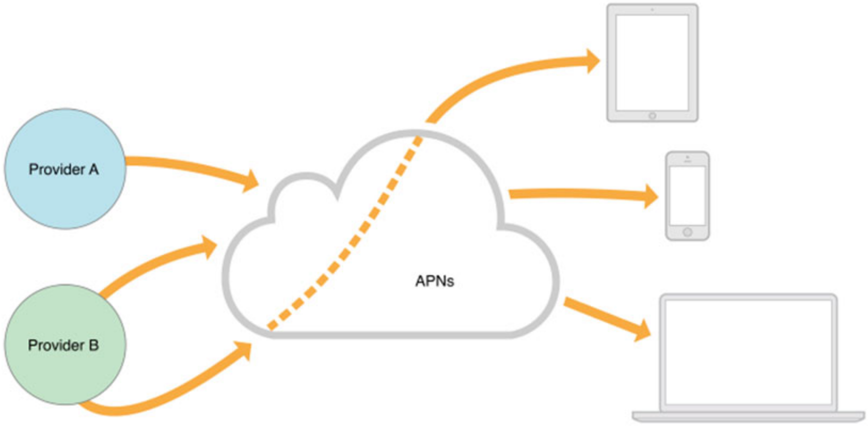
Allow apps to refresh their content when on Wi-Fi or cellular in the background. Turning off apps may help preserve battery life.

App Store
Apple Store
Apple Support
Books
Maps
News
Notes
Podcasts
Stocks
Voice Memos
Weather

https://support.apple.com/en-us/HT202070

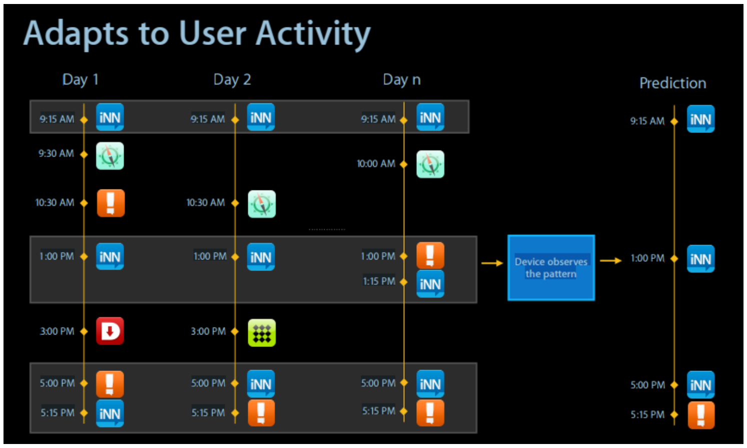
233. Claim 1 of the ’029 patent recites “communicating over the established multiplexed connection.” The ’029 accused products satisfy this limitation. For example, the ’029 accused products are configured for communicating over the established multiplexed connection, as shown in the exemplary documentation below.

Figure 6-2 Pushing remote notifications from multiple providers to multiple devices

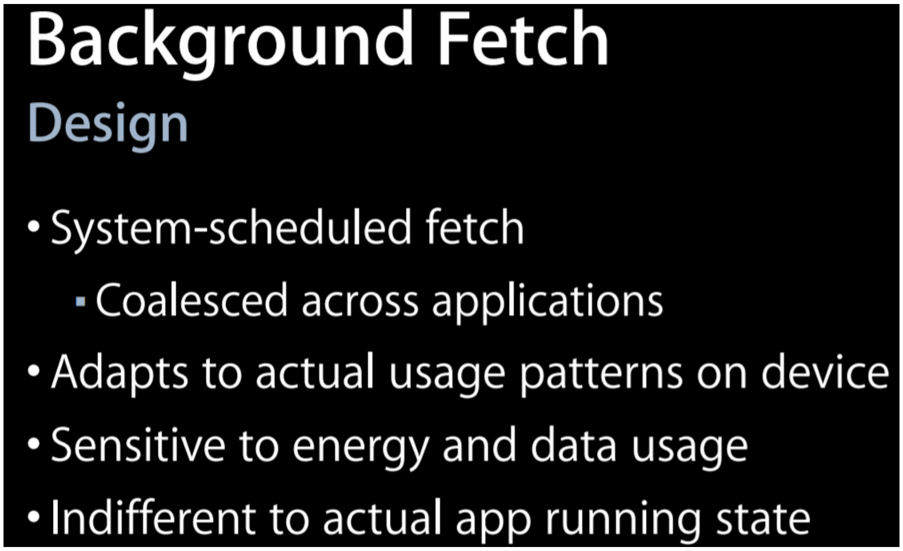


https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

234. Claim 1 of the ’029 patent recites “predicting an activity session based on application access history, wherein the application access history includes historical application usage.” The ’029 accused products satisfy this limitation. For example, the ’029 accused products are configured for predicting an activity session based on application access history, wherein the application access history includes historical application usage, as shown by the exemplary documentation and developer conference videos below.


Adapts to User Activity

https://developer.apple.com/videos/wwdc/2013/ (“What’s New with Multitasking”)

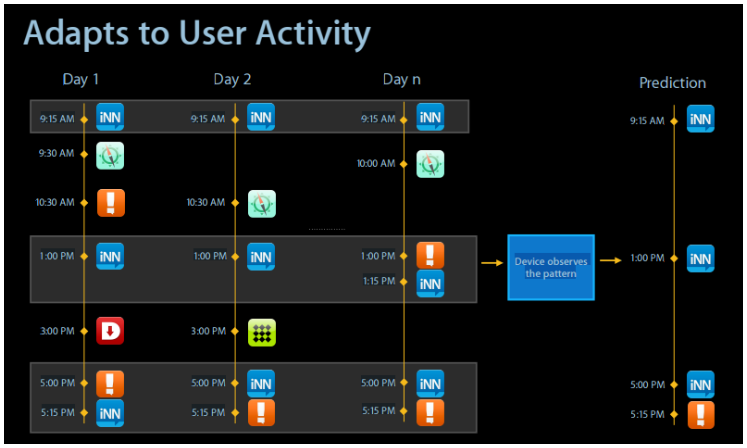

Background Fetch

Design

- System-scheduled fetch
  - Coalesced across applications
- Adapts to actual usage patterns on device
- Sensitive to energy and data usage
- Indifferent to actual app running state

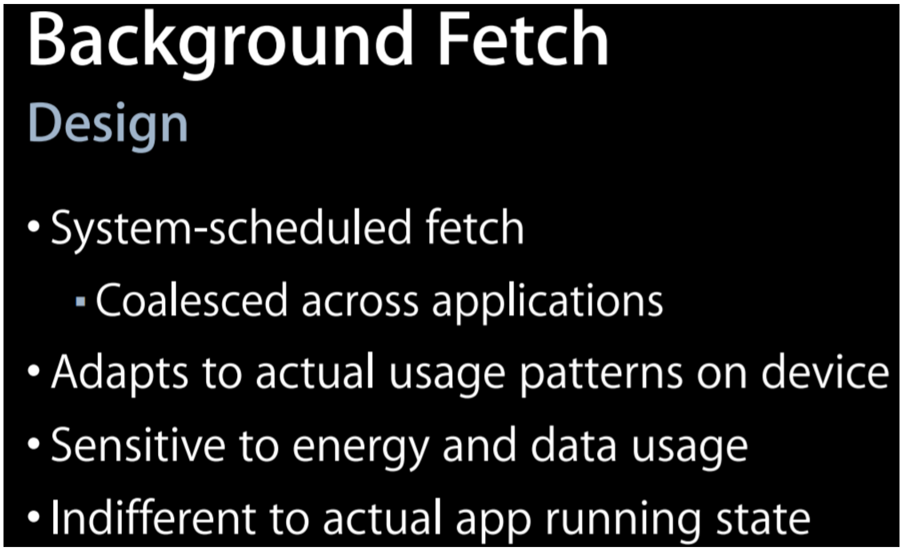
https://developer.apple.com/videos/wwdc/2013/ (“What’s New with Multitasking”)

235. Claim 1 of the ’029 patent recites “fetching data for the application before the activity session to support the predicted activity session before beginning the activity session.” The ’029 accused products satisfy this limitation. For example, the ’029 accused products are configured for fetching data for the application before the activity session to support the predicted activity session before beginning the activity session, as shown by the exemplary documentation and developer conference videos below.



Adapts to User Activity

https://developer.apple.com/videos/wwdc/2013/ (“What’s New with Multitasking”)



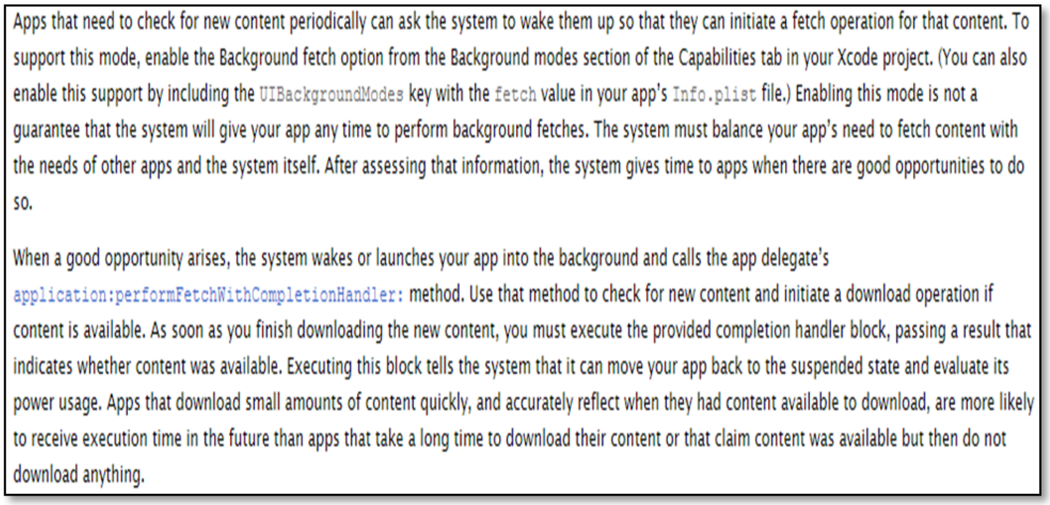
Background Fetch

Design

- System-scheduled fetch
  - Coalesced across applications
- Adapts to actual usage patterns on device
- Sensitive to energy and data usage
- Indifferent to actual app running state

https://developer.apple.com/videos/wwdc/2013/ (“What’s New with Multitasking”)

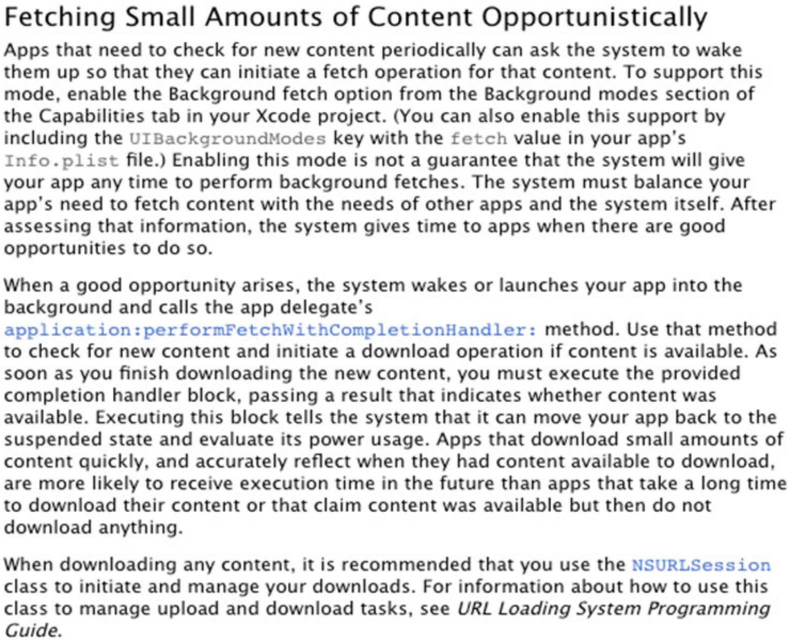
236. Claim 1 of the ’029 patent recites “wherein the application is operating in a background of the mobile device.” The ’029 accused products satisfy this limitation. For example, the application is operating in a background of the mobile device. For example, Background Fetch is a background execution mode and only operates while the app is in the background, as shown by the exemplary documentation below



Apps that need to check for new content periodically can ask the system to wake them up so that they can initiate a fetch operation for that content. To support this mode, enable the Background fetch option from the Background modes section of the Capabilities tab in your Xcode project. (You can also enable this support by including the `UIBackgroundModes` key with the `fetch` value in your app’s `Info.plist` file.) Enabling this mode is not a guarantee that the system will give your app any time to perform background fetches. The system must balance your app’s need to fetch content with the needs of other apps and the system itself. After assessing that information, the system gives time to apps when there are good opportunities to do so.

When a good opportunity arises, the system wakes or launches your app into the background and calls the app delegate’s `application:performFetchWithCompletionHandler:` method. Use that method to check for new content and initiate a download operation if content is available. As soon as you finish downloading the new content, you must execute the provided completion handler block, passing a result that indicates whether content was available. Executing this block tells the system that it can move your app back to the suspended state and evaluate its power usage. Apps that download small amounts of content quickly, and accurately reflect when they had content available to download, are more likely to receive execution time in the future than apps that take a long time to download their content or that claim content was available but then do not download anything.

https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html



Fetching Small Amounts of Content Opportunistically

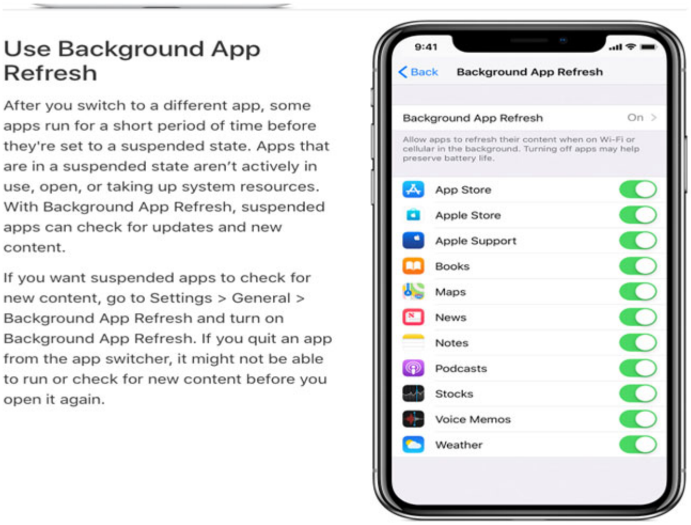
Apps that need to check for new content periodically can ask the system to wake them up so that they can initiate a fetch operation for that content. To support this mode, enable the Background fetch option from the Background modes section of the Capabilities tab in your Xcode project. (You can also enable this support by including the `UIBackgroundModes` key with the `fetch` value in your app’s `Info.plist` file.) Enabling this mode is not a guarantee that the system will give your app any time to perform background fetches. The system must balance your app’s need to fetch content with the needs of other apps and the system itself. After assessing that information, the system gives time to apps when there are good opportunities to do so.

When a good opportunity arises, the system wakes or launches your app into the background and calls the app delegate’s `application:performFetchWithCompletionHandler:` method. Use that method to check for new content and initiate a download operation if content is available. As soon as you finish downloading the new content, you must execute the provided completion handler block, passing a result that indicates whether content was available. Executing this block tells the system that it can move your app back to the suspended state and evaluate its power usage. Apps that download small amounts of content quickly, and accurately reflect when they had content available to download, are more likely to receive execution time in the future than apps that take a long time to download their content or that claim content was available but then do not download anything.

When downloading any content, it is recommended that you use the `NSURLSession` class to initiate and manage your downloads. For information about how to use this class to manage upload and download tasks, see *URL Loading System Programming Guide*.

https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html#//apple_ref/doc/uid/TP40007072-CH4-SW56

237. Claim 1 of the ’029 patent recites “wherein the data is fetched if the fetching is enabled by the user selection for the application.” The ’029 accused products satisfy this limitation. For example, data is fetched if the fetching is enabled by the user selection for the application, as shown in the exemplary documentation below.



Use Background App Refresh

After you switch to a different app, some apps run for a short period of time before they're set to a suspended state. Apps that are in a suspended state aren't actively in use, open, or taking up system resources. With Background App Refresh, suspended apps can check for updates and new content.

If you want suspended apps to check for new content, go to Settings > General > Background App Refresh and turn on Background App Refresh. If you quit an app from the app switcher, it might not be able to run or check for new content before you open it again.

9:41

< Back Background App Refresh

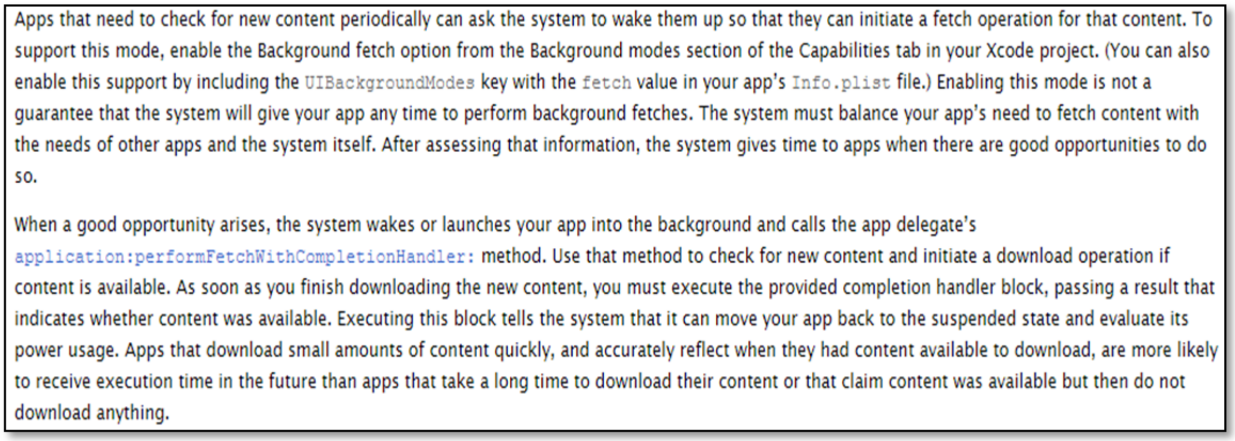
Background App Refresh On >

Allow apps to refresh their content when on Wi-Fi or cellular in the background. Turning off apps may help preserve battery life.

App Store
Apple Store
Apple Support
Books
Maps
News
Notes
Podcasts
Stocks
Voice Memos
Weather

https://support.apple.com/en-us/HT202070

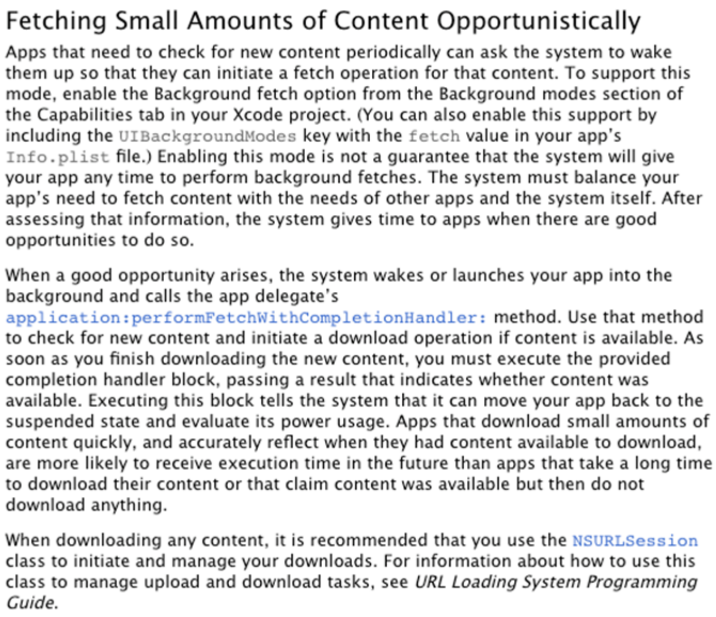
238. Claim 1 of the ’029 patent recites “wherein at least some of the fetched data is for background requests made by the application on the mobile device.” The ’029 accused products satisfy this limitation. For example, ’029 accused products fetch data for background requests made by the application on the mobile device, as shown in the exemplary documentation below.



Apps that need to check for new content periodically can ask the system to wake them up so that they can initiate a fetch operation for that content. To support this mode, enable the Background fetch option from the Background modes section of the Capabilities tab in your Xcode project. (You can also enable this support by including the `UIBackgroundModes` key with the `fetch` value in your app's `Info.plist` file.) Enabling this mode is not a guarantee that the system will give your app any time to perform background fetches. The system must balance your app's need to fetch content with the needs of other apps and the system itself. After assessing that information, the system gives time to apps when there are good opportunities to do so.

When a good opportunity arises, the system wakes or launches your app into the background and calls the app delegate's `application:performFetchWithCompletionHandler:` method. Use that method to check for new content and initiate a download operation if content is available. As soon as you finish downloading the new content, you must execute the provided completion handler block, passing a result that indicates whether content was available. Executing this block tells the system that it can move your app back to the suspended state and evaluate its power usage. Apps that download small amounts of content quickly, and accurately reflect when they had content available to download, are more likely to receive execution time in the future than apps that take a long time to download their content or that claim content was available but then do not download anything.

https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html



Fetching Small Amounts of Content Opportunistically

Apps that need to check for new content periodically can ask the system to wake them up so that they can initiate a fetch operation for that content. To support this mode, enable the Background fetch option from the Background modes section of the Capabilities tab in your Xcode project. (You can also enable this support by including the `UIBackgroundModes` key with the `fetch` value in your app's `Info.plist` file.) Enabling this mode is not a guarantee that the system will give your app any time to perform background fetches. The system must balance your app's need to fetch content with the needs of other apps and the system itself. After assessing that information, the system gives time to apps when there are good opportunities to do so.

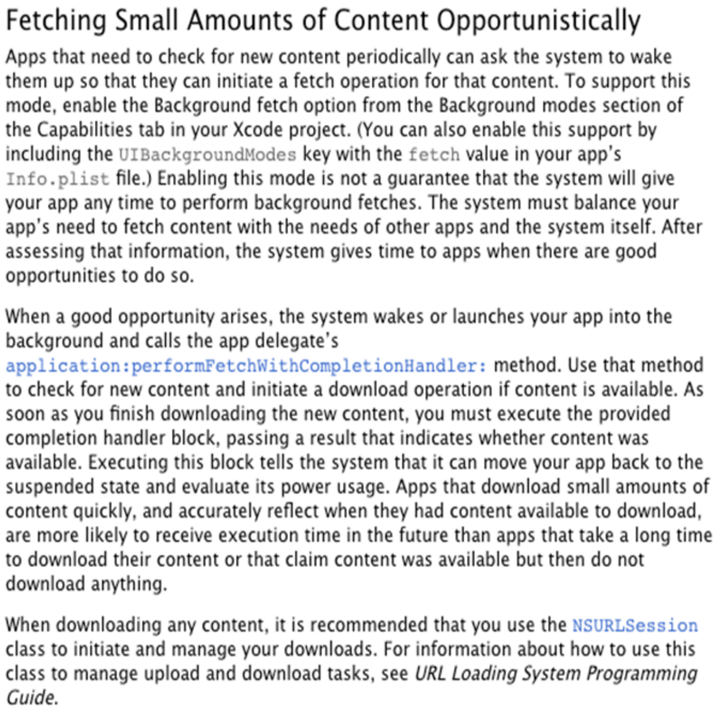
When a good opportunity arises, the system wakes or launches your app into the background and calls the app delegate's `application:performFetchWithCompletionHandler:` method. Use that method to check for new content and initiate a download operation if content is available. As soon as you finish downloading the new content, you must execute the provided completion handler block, passing a result that indicates whether content was available. Executing this block tells the system that it can move your app back to the suspended state and evaluate its power usage. Apps that download small amounts of content quickly, and accurately reflect when they had content available to download, are more likely to receive execution time in the future than apps that take a long time to download their content or that claim content was available but then do not download anything.

When downloading any content, it is recommended that you use the `NSURLSession` class to initiate and manage your downloads. For information about how to use this class to manage upload and download tasks, see *URL Loading System Programming Guide*.

https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html#//apple_ref/doc/uid/TP40007072-CH4-SW56

239. Claim 1 of the ’029 patent recites “wherein a second connection is established that is other than the established multiplexed connection with the mobile device, wherein fetching data occurs over the second connection; and disconnecting from the second connection.” The ’029 accused products satisfy this limitation. For example, the ’029 accused

products establish a second connection, other than the established connection with the mobile device, and fetch data over the second connection. The second connection is also disconnected. Background fetch is performed to application servers, for example using NSURLSession, which establishes a connection separate from the APNs connection, and NSURLSession connection is terminated after the background fetch is complete, as shown in the exemplary documentation below.



Fetching Small Amounts of Content Opportunistically

Apps that need to check for new content periodically can ask the system to wake them up so that they can initiate a fetch operation for that content. To support this mode, enable the Background fetch option from the Background modes section of the Capabilities tab in your Xcode project. (You can also enable this support by including the `UIBackgroundModes` key with the `fetch` value in your app's `Info.plist` file.) Enabling this mode is not a guarantee that the system will give your app any time to perform background fetches. The system must balance your app's need to fetch content with the needs of other apps and the system itself. After assessing that information, the system gives time to apps when there are good opportunities to do so.

When a good opportunity arises, the system wakes or launches your app into the background and calls the app delegate's `application:performFetchWithCompletionHandler:` method. Use that method to check for new content and initiate a download operation if content is available. As soon as you finish downloading the new content, you must execute the provided completion handler block, passing a result that indicates whether content was available. Executing this block tells the system that it can move your app back to the suspended state and evaluate its power usage. Apps that download small amounts of content quickly, and accurately reflect when they had content available to download, are more likely to receive execution time in the future than apps that take a long time to download their content or that claim content was available but then do not download anything.

When downloading any content, it is recommended that you use the `NSURLSession` class to initiate and manage your downloads. For information about how to use this class to manage upload and download tasks, see *URL Loading System Programming Guide*.

https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html#//apple_ref/doc/uid/TP40007072-CH4-SW56

240. Thus, as illustrated above, the '029 accused products directly infringe one or more claims of the '029 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the '029 patent.

241. Apple indirectly infringes the ’029 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple’s customers and end-users, in this district and elsewhere in the United States. For example, Apple’s customers and end-users directly infringe through their use of the inventions claimed in the ’029 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’029 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the ’029 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products’ capability in an infringing manner, including through the use of Apple’s background app data fetching and background app refresh. *See, e.g.*, https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html; https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html#//apple_ref/doc/uid/TP40007072-CH4-SW56; https://support.apple.com/en-us/HT202070. As a result of Apple’s inducement, Apple’s customers and end-users use the Apple products in the way Apple intends and directly infringe the ’029 patent. Apple performs these affirmative acts with knowledge of the ’029

patent and with the intent, or willful blindness, that the induced acts directly infringe the ’029 patent. Apple has had knowledge and notice of the ’029 patent at least as of the filing of this complaint.

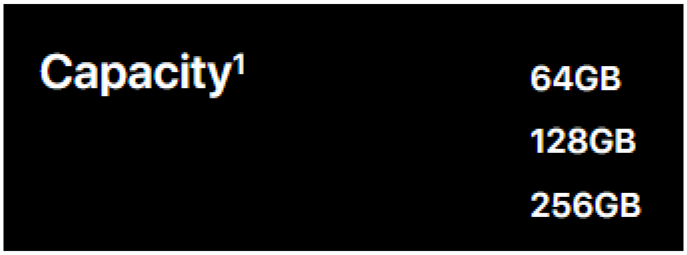
242. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

COUNT XIII: PATENT INFRINGEMENT OF THE ’734 PATENT

243. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

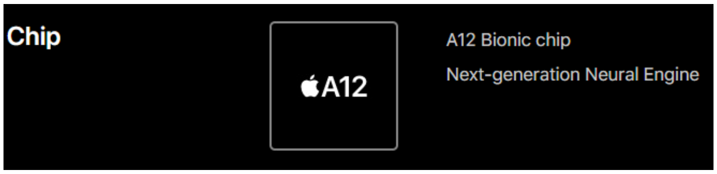
244. Apple infringes (literally and/or under the Doctrine of Equivalents) the ’734 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the ’734 patent including at least the ’734 accused products noted above. In addition, Apple derives revenue from the activities relating to the ’734 accused products.

245. For example and as shown below, the ’734 accused products infringe at least claim 1 of the ’734 patent. For example, to the extent the preamble is limiting, the ’734 accused products satisfy a “mobile device which improves network resource utilization in a wireless network, the mobile device, comprising a memory; a radio; and a processor coupled to the memory and configured to:” As shown below by the exemplary evidence, the ’734 accused products satisfy a mobile device which improves network resource utilization in a wireless network, the mobile device comprising a memory, a radio, and a processor, as shown by the exemplary documentation below. These devices include, for example, Apple’s Low Power Mode, which improves network resource utilization by, among other things, reducing background traffic.



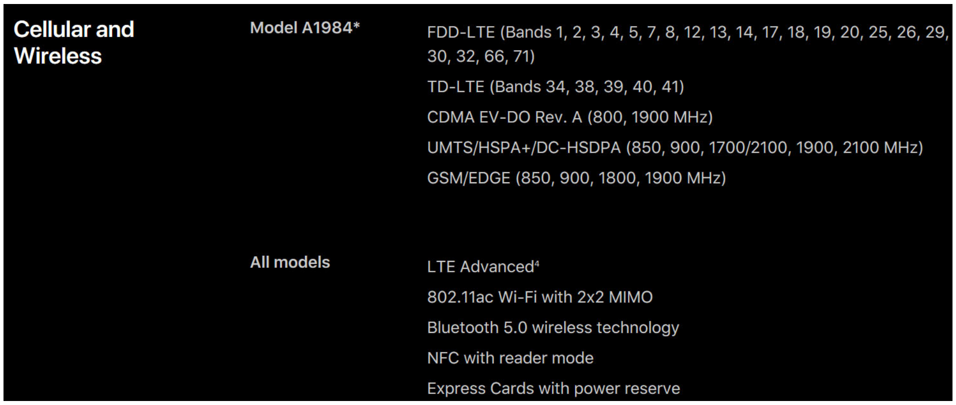
Capacity[1]	
	64GB 128GB 256GB

https://www.apple.com/iphone-xr/specs/



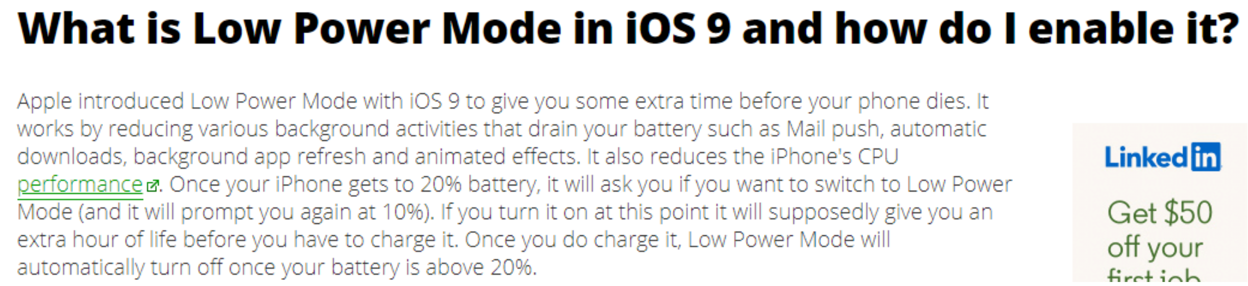
Chip	A12	A12 Bionic chip Next-generation Neural Engine

https://www.apple.com/iphone-xr/specs/



Cellular and Wireless		
	Model A1984*	FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41) CDMA EV-DO Rev. A (800, 1900 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
	All models	LTE Advanced[4] 802.11ac Wi-Fi with 2x2 MIMO Bluetooth 5.0 wireless technology NFC with reader mode Express Cards with power reserve

https://www.apple.com/iphone-xr/specs/

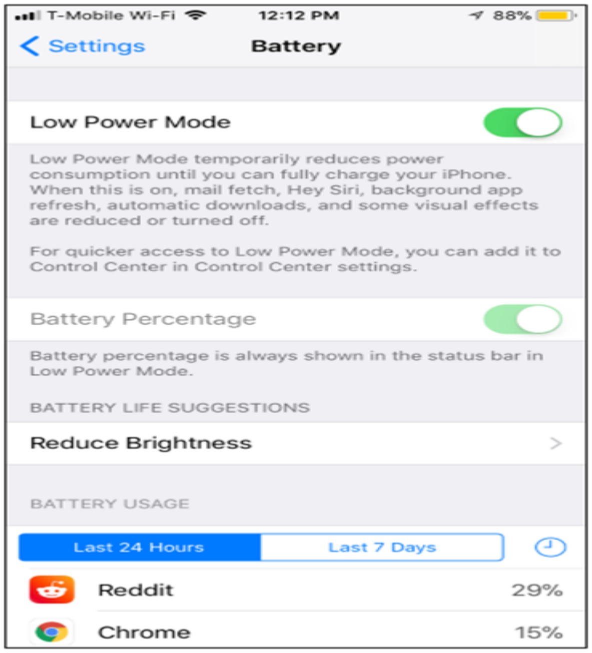


What is Low Power Mode in iOS 9 and how do I enable it?

Apple introduced Low Power Mode with iOS 9 to give you some extra time before your phone dies. It works by reducing various background activities that drain your battery such as Mail push, automatic downloads, background app refresh and animated effects. It also reduces the iPhone's CPU performance. Once your iPhone gets to 20% battery, it will ask you if you want to switch to Low Power Mode (and it will prompt you again at 10%). If you turn it on at this point it will supposedly give you an extra hour of life before you have to charge it. Once you do charge it, Low Power Mode will automatically turn off once your battery is above 20%.

https://www.iphonefaq.org/archives/974871

246. Claim 1 of the ’734 patent recites “receive instructions from a user to enter a power save mode.” The ’734 accused products satisfy this limitation. For example, the ’734 accused products receive instructions from a user to enter power save mode, as shown by the exemplary documentation below.



Battery

Low Power Mode

Low Power Mode temporarily reduces power consumption until you can fully charge your iPhone. When this is on, mail fetch, Hey Siri, background app refresh, automatic downloads, and some visual effects are reduced or turned off.

For quicker access to Low Power Mode, you can add it to Control Center in Control Center settings.

Battery Percentage

Battery percentage is always shown in the status bar in Low Power Mode.

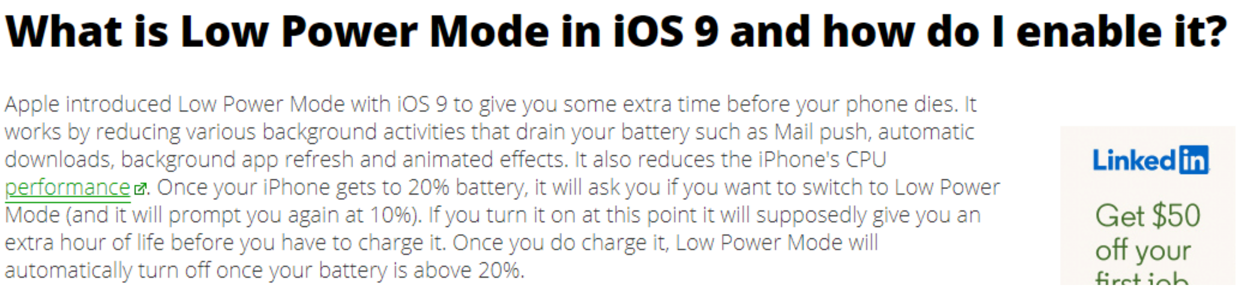
BATTERY LIFE SUGGESTIONS

Reduce Brightness >

BATTERY USAGE

Last 24 Hours | Last 7 Days

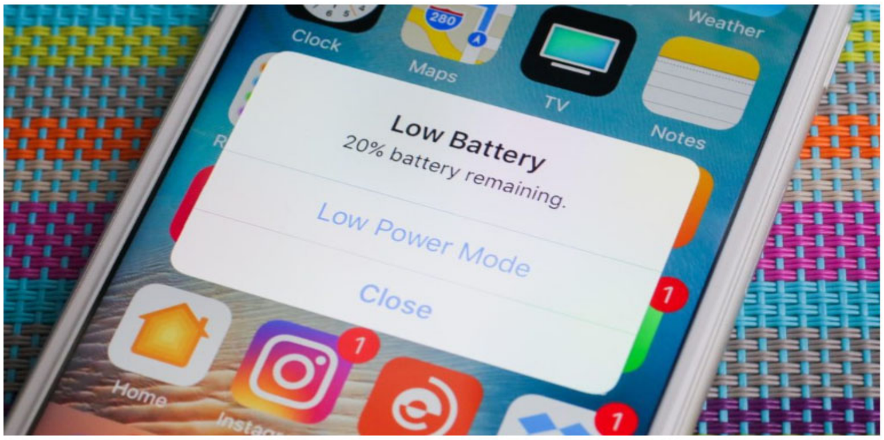
Reddit	29%
Chrome	15%



What is Low Power Mode in iOS 9 and how do I enable it?

Apple introduced Low Power Mode with iOS 9 to give you some extra time before your phone dies. It works by reducing various background activities that drain your battery such as Mail push, automatic downloads, background app refresh and animated effects. It also reduces the iPhone's CPU performance. Once your iPhone gets to 20% battery, it will ask you if you want to switch to Low Power Mode (and it will prompt you again at 10%). If you turn it on at this point it will supposedly give you an extra hour of life before you have to charge it. Once you do charge it, Low Power Mode will automatically turn off once your battery is above 20%.

https://www.iphonefaq.org/archives/974871

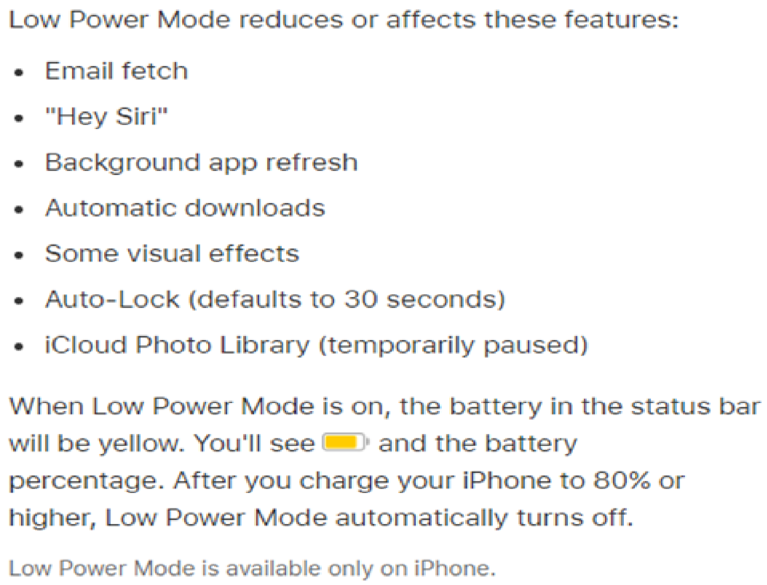


Low Battery
20% battery remaining.
Low Power Mode
Close

https://www.thefonestuff.com/blog/how-to-improve-iphone-x-battery-life/

247. Claim 1 of the ’734 patent recites “while in the power save mode, block transmission of outgoing application data requests, wherein the outgoing application data

requests are background application requests for more than one application." The '734 accused products satisfy this limitation. For example, in low power mode, the '734 accused products block application requests for more than one application, as shown in the exemplary documentation below.



Low Power Mode reduces or affects these features:

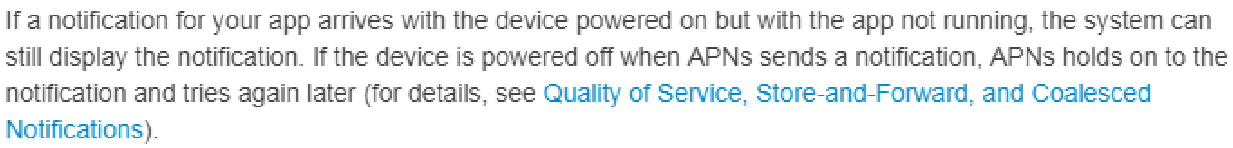
- Email fetch
- "Hey Siri"
- Background app refresh
- Automatic downloads
- Some visual effects
- Auto-Lock (defaults to 30 seconds)
- iCloud Photo Library (temporarily paused)

When Low Power Mode is on, the battery in the status bar will be yellow. You'll see 🔋 and the battery percentage. After you charge your iPhone to 80% or higher, Low Power Mode automatically turns off.

Low Power Mode is available only on iPhone.

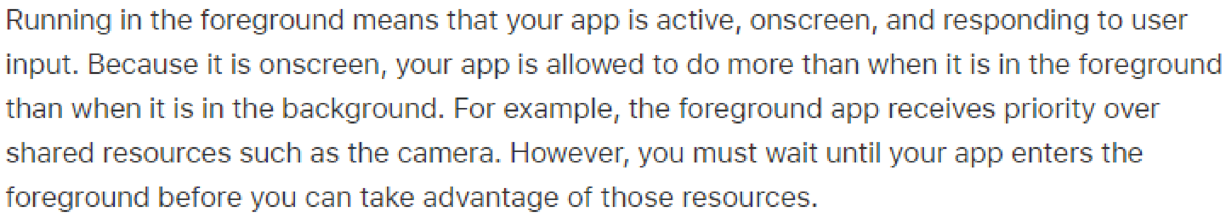
https://support.apple.com/en-us/HT205234

248. Claim 1 of the '734 patent recites "while in the power save mode, allow transmission of additional outgoing application data requests in response to occurrence of receipt of data transfer from a remote entity, user input in response to a prompt displayed to the user, and a change in a background status of an application executing on the mobile device, wherein the additional outgoing application data requests are foreground application requests." The '734 accused products satisfy this limitation. For example, in low power mode, the '734 accused products allow transmission of foreground app data requests in response to messages from Apple APNs, user input, and moving an app from background to foreground, as shown in the exemplary documentation below.



If a notification for your app arrives with the device powered on but with the app not running, the system can still display the notification. If the device is powered off when APNs sends a notification, APNs holds on to the notification and tries again later (for details, see Quality of Service, Store-and-Forward, and Coalesced Notifications).

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

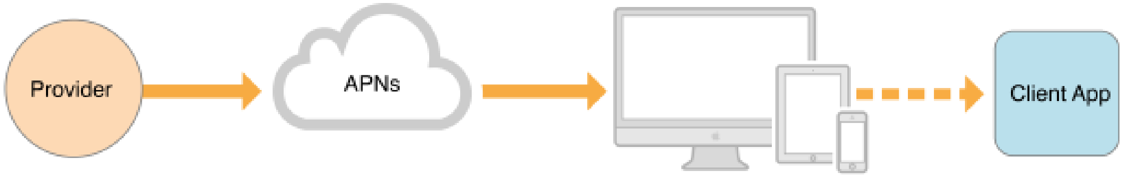


Running in the foreground means that your app is active, onscreen, and responding to user input. Because it is onscreen, your app is allowed to do more than when it is in the foreground than when it is in the background. For example, the foreground app receives priority over shared resources such as the camera. However, you must wait until your app enters the foreground before you can take advantage of those resources.

https://developer.apple.com/documentation/uikit/core_app/managing_your_app_s_life_cycle/preparing_your_app_to_run_in_the_foreground?language=objc

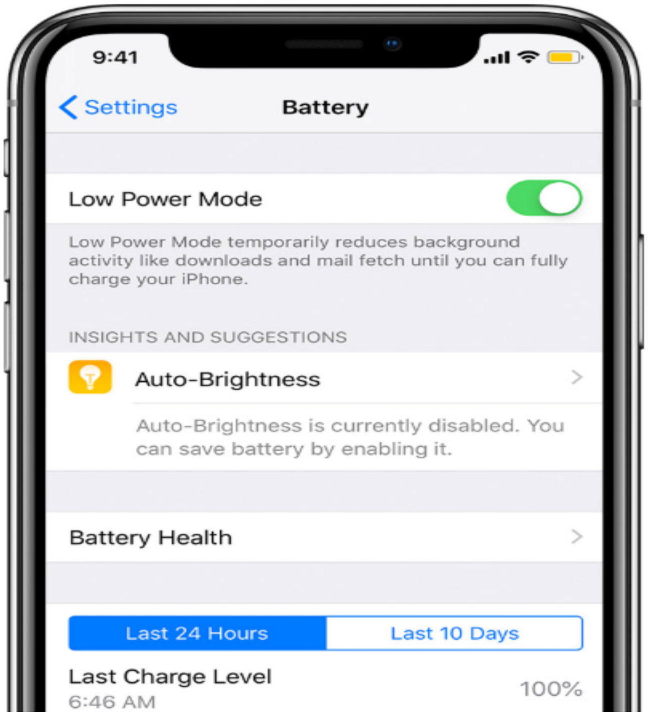
249. Claim 1 of the ’734 patent recites “wherein the remote entity is an intermediary server that provides connectivity between an application server and the application and the mobile device.” The ’734 accused products satisfy this limitation. For example, Apple Push Notification service (APNs) includes an intermediary server that provides connectivity between an app server and the ’734 accused products, as shown in the exemplary documentation below.

Figure 6-1 Delivering a remote notification from a provider to an app



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

250. Claim 1 of the ’734 patent recites “exit the power save mode based on received instructions from the user to exit the power save mode.” The ’734 accused products satisfy this limitation. For example, the ’734 accused products exit low power mode when the user turns it off.



9:41

< Settings Battery

Low Power Mode

Low Power Mode temporarily reduces background activity like downloads and mail fetch until you can fully charge your iPhone.

INSIGHTS AND SUGGESTIONS

Auto-Brightness >

Auto-Brightness is currently disabled. You can save battery by enabling it.

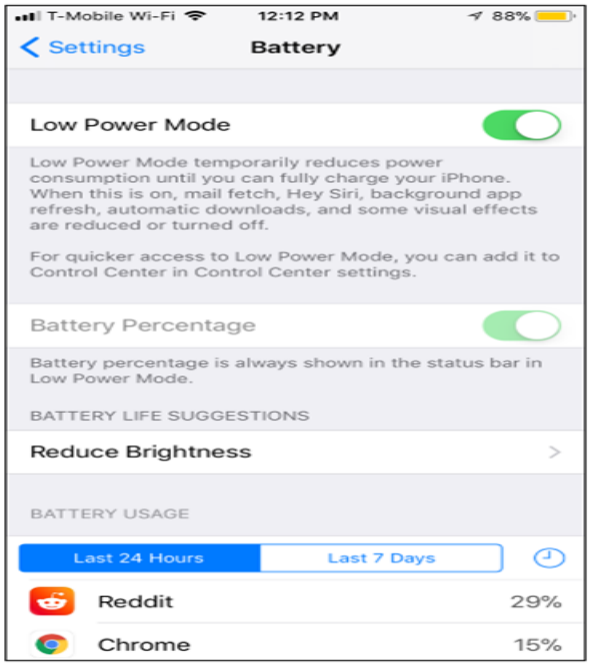
Battery Health >

Last 24 Hours | Last 10 Days

Last Charge Level 100%

6:46 AM

https://support.apple.com/en-us/HT205234



T-Mobile Wi-Fi 12:12 PM 88%

< Settings Battery

Low Power Mode

Low Power Mode temporarily reduces power consumption until you can fully charge your iPhone. When this is on, mail fetch, Hey Siri, background app refresh, automatic downloads, and some visual effects are reduced or turned off.

For quicker access to Low Power Mode, you can add it to Control Center in Control Center settings.

Battery Percentage

Battery percentage is always shown in the status bar in Low Power Mode.

BATTERY LIFE SUGGESTIONS

Reduce Brightness >

BATTERY USAGE

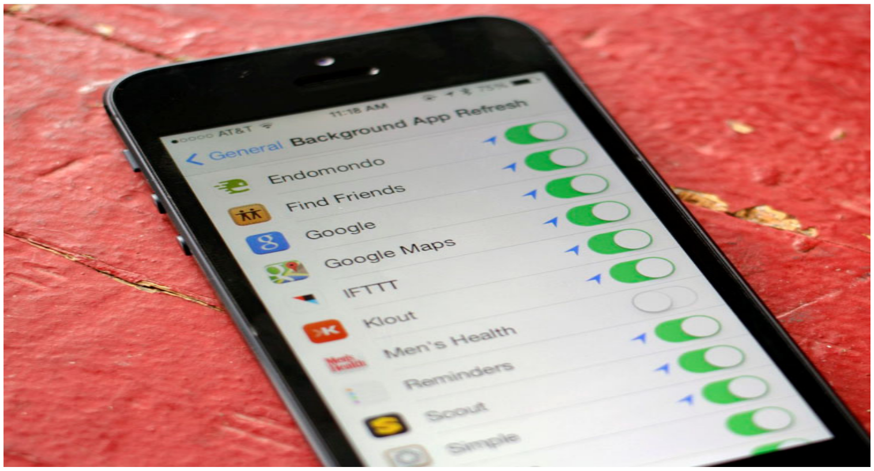
Last 24 Hours | Last 7 Days

Reddit 29%

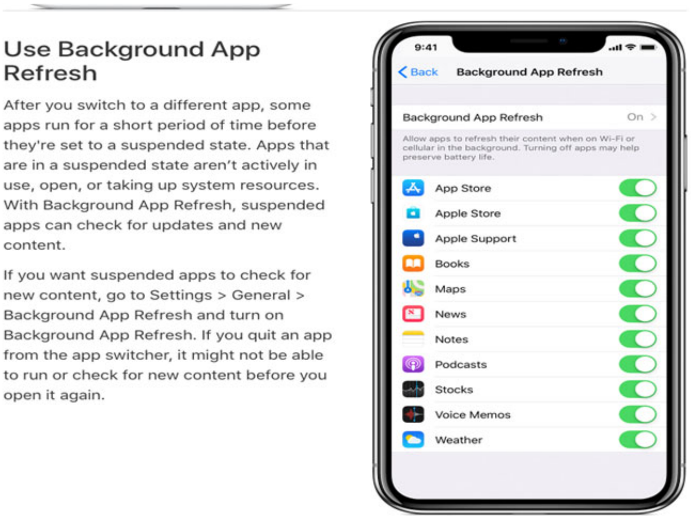
Chrome 15%

251. Claim 1 of the ’734 patent recites “wherein, when the power save mode is exited, the outgoing application data requests occurring while the mobile device is not in power save mode are blocked by user selection on an application-by-application basis, wherein the user

selection instructs the mobile device whether to block the outgoing application data requests for each application that is selected by the user for blocking." The '734 accused products satisfy this limitation. For example, the user can configure background app refresh on an app-by-app basis when not in low power mode, which blocks outgoing app data requests.



https://www.cnet.com/how-to/how-to-turn-off-background-app-refresh-in-ios-7/



Use Background App Refresh

After you switch to a different app, some apps run for a short period of time before they're set to a suspended state. Apps that are in a suspended state aren't actively in use, open, or taking up system resources. With Background App Refresh, suspended apps can check for updates and new content.

If you want suspended apps to check for new content, go to Settings > General > Background App Refresh and turn on Background App Refresh. If you quit an app from the app switcher, it might not be able to run or check for new content before you open it again.

9:41

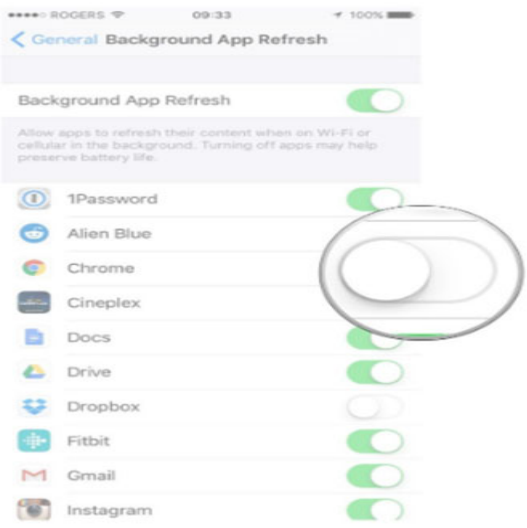
< Back Background App Refresh

Background App Refresh On >

Allow apps to refresh their content when on Wi-Fi or cellular in the background. Turning off apps may help preserve battery life.

App Store
Apple Store
Apple Support
Books
Maps
News
Notes
Podcasts
Stocks
Voice Memos
Weather

https://support.apple.com/en-us/HT202070



Background App Refresh

Allow apps to refresh their content when on Wi-Fi or cellular in the background. Turning off apps may help preserve battery life.

1Password
Alien Blue
Chrome
Cineplex
Docs
Drive
Dropbox
Fitbit
Gmail
Instagram

252. Thus, as illustrated above, the ’734 accused products directly infringe one or more claims of the ’734 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the ’734 patent.

253. Apple indirectly infringes the ’734 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple’s customers and end-users, in this district and elsewhere in the United States. For example, Apple’s customers and end-users directly infringe through their use of the inventions claimed in the ’734 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’734 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the ’734 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of

infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products' capability in an infringing manner, including including through the use of Apple's low power mode. *See, e.g.,* https://support.apple.com/en-us/HT205234. As a result of Apple's inducement, Apple's customers and end-users use the Apple products in the way Apple intends and directly infringe the '734 patent. Apple performs these affirmative acts with knowledge of the '734 patent and with the intent, or willful blindness, that the induced acts directly infringe the '734 patent. Apple has had knowledge and notice of the '734 patent at least as of the filing of this complaint.

254. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

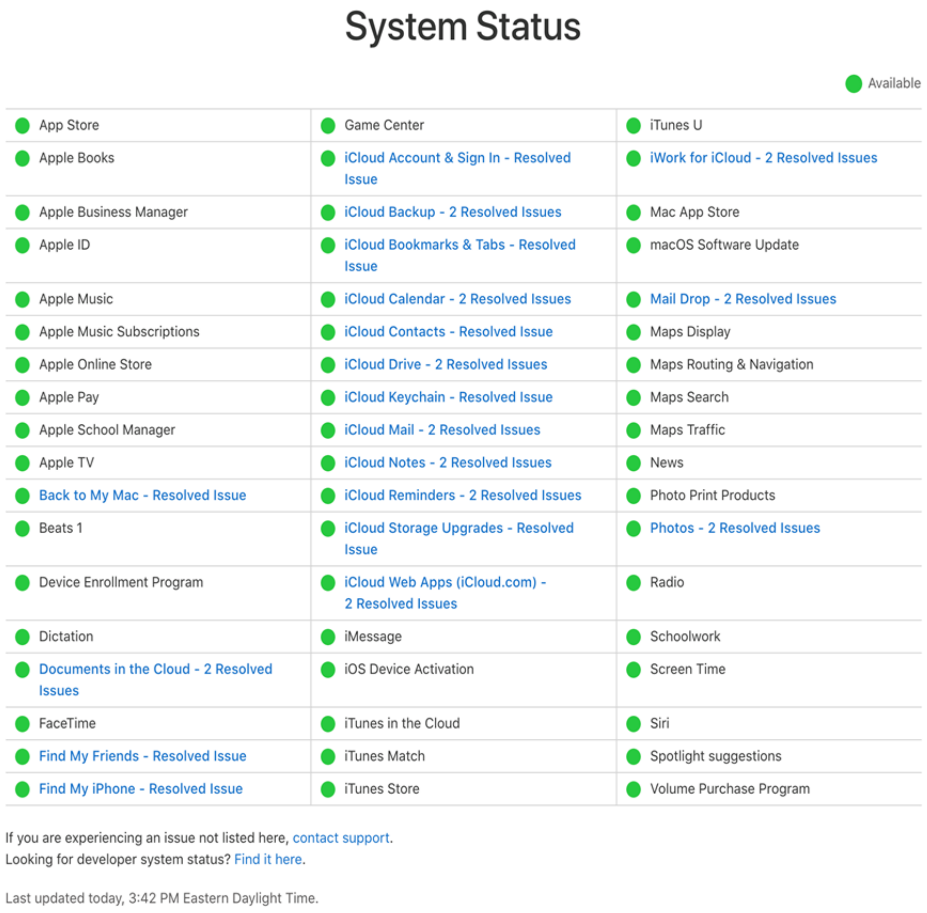
COUNT XIV: PATENT INFRINGEMENT OF THE '534 PATENT

255. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

256. Apple infringes (literally and/or under the Doctrine of Equivalents) the '534 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the '534 patent including at least the '534 accused products noted above. In addition, Apple derives revenue from the activities relating to the '534 accused products.

257. For example and as shown below, the '534 accused products infringe at least claim 1 of the '534 patent. For example, to the extent the preamble is limiting, the '534 accused products satisfy a "server that manages transactions between first and second devices, the server comprising: a communications interface; a processor communicatively coupled to the communication interface; and a memory communicatively coupled to the processor, the memory

containing instructions executable by the processor, whereby the server is operable to:" As shown below by the exemplary evidence, Apple's server that supports, for example, Find My Friends and Find My iPhone applications, manages transactions between first and second devices (e.g., iPhones, iPads, macs).[5]



System Status

● Available

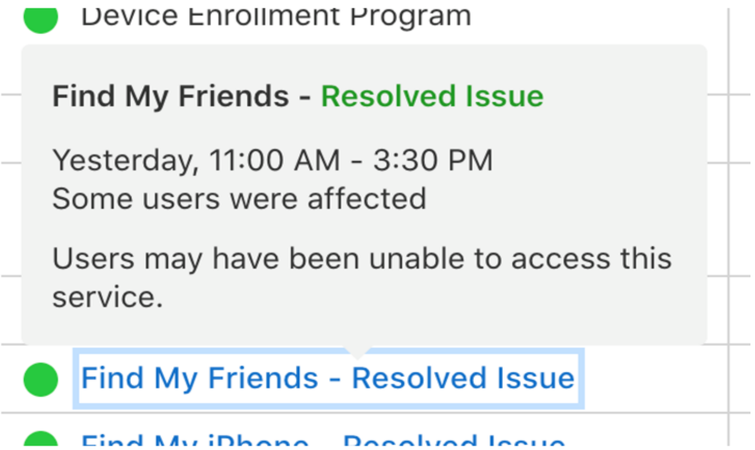
● App Store	● Game Center	● iTunes U
● Apple Books	● iCloud Account & Sign In - Resolved Issue	● iWork for iCloud - 2 Resolved Issues
● Apple Business Manager	● iCloud Backup - 2 Resolved Issues	● Mac App Store
● Apple ID	● iCloud Bookmarks & Tabs - Resolved Issue	● macOS Software Update
● Apple Music	● iCloud Calendar - 2 Resolved Issues	● Mail Drop - 2 Resolved Issues
● Apple Music Subscriptions	● iCloud Contacts - Resolved Issue	● Maps Display
● Apple Online Store	● iCloud Drive - 2 Resolved Issues	● Maps Routing & Navigation
● Apple Pay	● iCloud Keychain - Resolved Issue	● Maps Search
● Apple School Manager	● iCloud Mail - 2 Resolved Issues	● Maps Traffic
● Apple TV	● iCloud Notes - 2 Resolved Issues	● News
● Back to My Mac - Resolved Issue	● iCloud Reminders - 2 Resolved Issues	● Photo Print Products
● Beats 1	● iCloud Storage Upgrades - Resolved Issue	● Photos - 2 Resolved Issues
● Device Enrollment Program	● iCloud Web Apps (iCloud.com) - 2 Resolved Issues	● Radio
● Dictation	● iMessage	● Schoolwork
● Documents in the Cloud - 2 Resolved Issues	● iOS Device Activation	● Screen Time
● FaceTime	● iTunes in the Cloud	● Siri
● Find My Friends - Resolved Issue	● iTunes Match	● Spotlight suggestions
● Find My iPhone - Resolved Issue	● iTunes Store	● Volume Purchase Program

If you are experiencing an issue not listed here, contact support.
Looking for developer system status? Find it here.

Last updated today, 3:42 PM Eastern Daylight Time.

https://www.apple.com/support/systemstatus/.

[5] With respect to Find my iPhone, for example, the first device may be either a mac or an iOS device such as an iPad, attempting to find the location of, for example, an iPhone (second device).



● Device Enrollment Program

Find My Friends - Resolved Issue

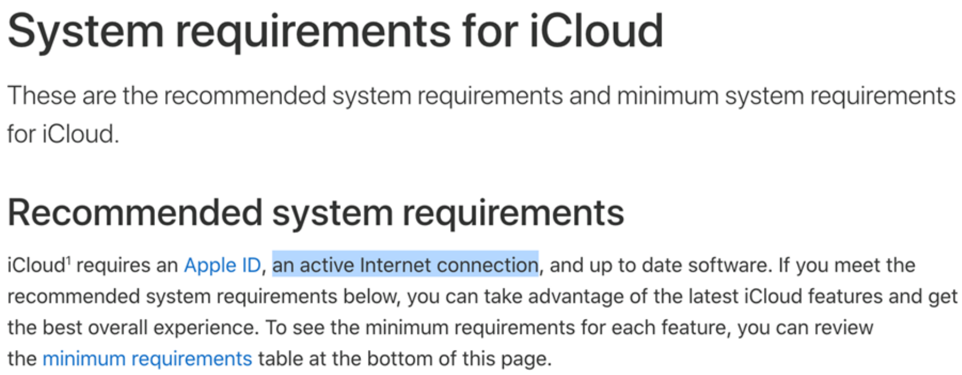
Yesterday, 11:00 AM - 3:30 PM
Some users were affected

Users may have been unable to access this service.

● Find My Friends - Resolved Issue

● Find My iPhone - Resolved Issue

https://www.apple.com/support/systemstatus/.



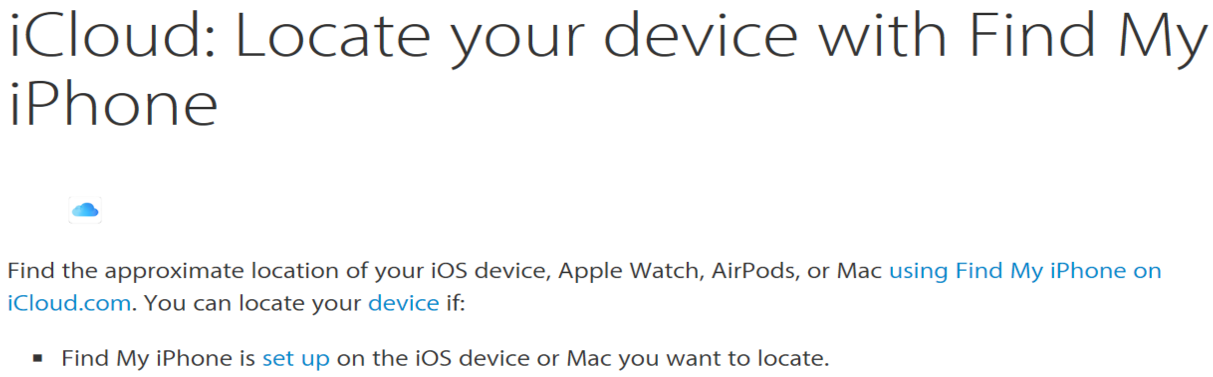
System requirements for iCloud

These are the recommended system requirements and minimum system requirements for iCloud.

Recommended system requirements

iCloud[1] requires an Apple ID, an active Internet connection, and up to date software. If you meet the recommended system requirements below, you can take advantage of the latest iCloud features and get the best overall experience. To see the minimum requirements for each feature, you can review the minimum requirements table at the bottom of this page.

https://support.apple.com/en-us/HT204230



iCloud: Locate your device with Find My iPhone

Find the approximate location of your iOS device, Apple Watch, AirPods, or Mac using Find My iPhone on iCloud.com. You can locate your device if:

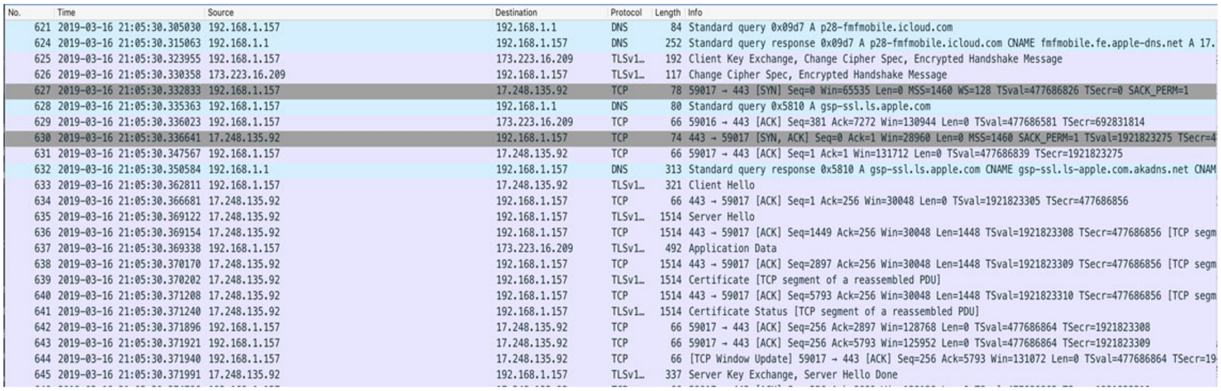
- Find My iPhone is set up on the iOS device or Mac you want to locate.

https://support.apple.com/kb/PH2698?viewlocale=en_US&locale=en_US

258. Claim 1 of the ’534 patent recites “receive a first connection associated with a first device.” The ’534 accused products receive a first connection associated with a first device. For example, the ’534 accused products receive a first connection from, for example, a first iPhone, as confirmed through the exemplary documentation below.

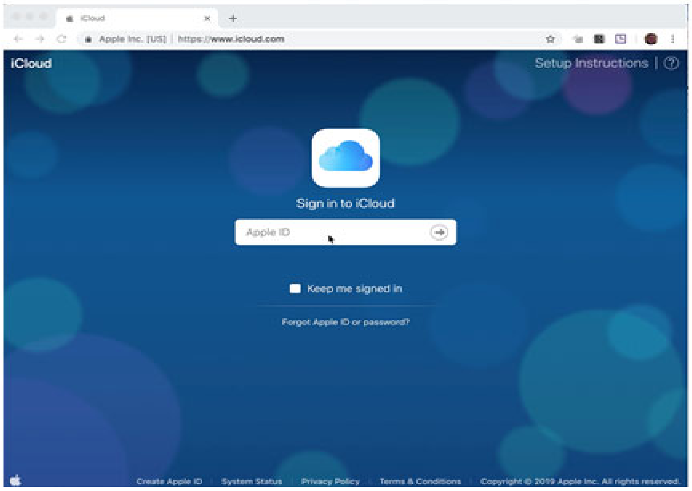
```
default  21:05:30.513956 -0400      FindMyFriends     TIC Enabling TLS [1:0x282535380]

default  21:05:30.513996 -0400      FindMyFriends     TIC TCP Conn Start [1:0x282535380]

default  21:05:30.514154 -0400      FindMyFriends     [C1     Hostname#78e5c4e8:443     tcp,     url:     https://p28-
fmfmobile.icloud.com/fmipservice/friends/1538423246/9387152d917ee7cc8ce9b3590b3cbe046fdad67d/first/initClien,     tls]
start

default  21:05:30.514434 -0400      FindMyFriends     nw_connection_report_state_with_handler_locked  [C1]  reporting
state preparing

default  21:05:30.517949 -0400      FindMyFriends     Task   <8254F4FE-AC65-4FCA-9020-A5FF5C0703F7>.<1>   setting   up
Connection 1
```

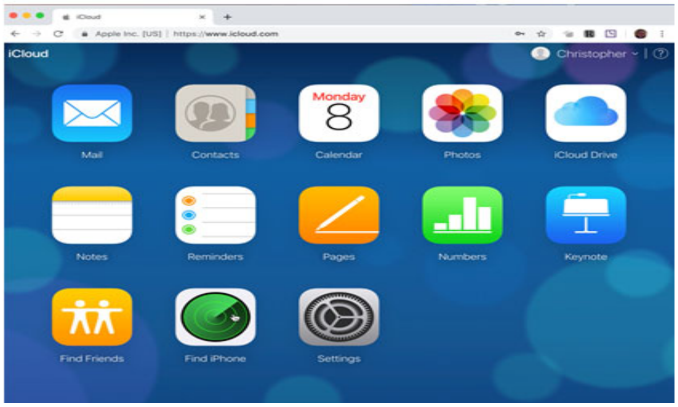
Log captured March 16, 2019

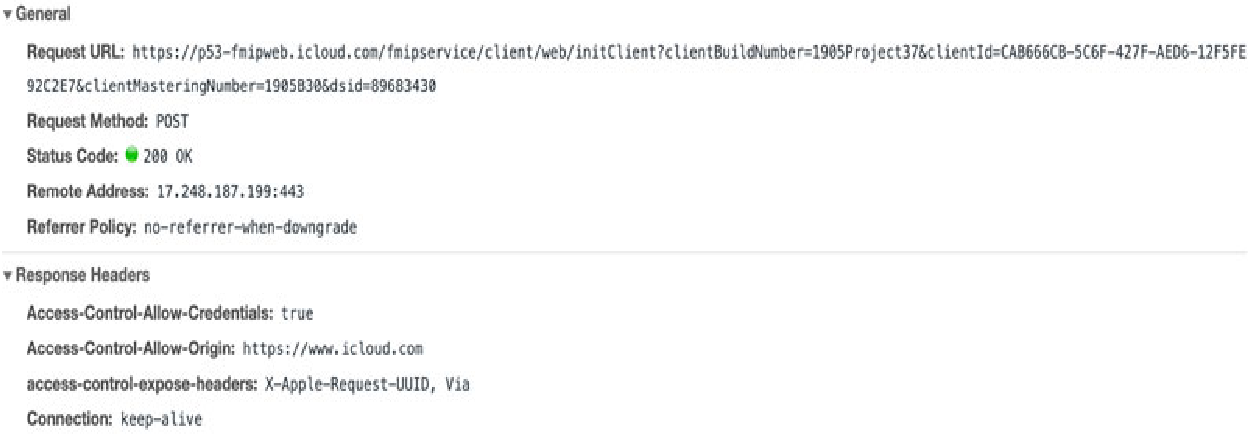


No.	Time	Source	Destination	Protocol	Length	Info
621	2019-03-16 21:05:30.305030	192.168.1.157	192.168.1.1	DNS	84	Standard query 0x09d7 A p28-fmfmobile.icloud.com
624	2019-03-16 21:05:30.315063	192.168.1.1	192.168.1.157	DNS	252	Standard query response 0x09d7 A p28-fmfmobile.icloud.com CNAME fmfmobile.fe.apple-dns.net A 17.
625	2019-03-16 21:05:30.323955	192.168.1.157	173.223.16.209	TLSv1…	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
626	2019-03-16 21:05:30.330358	173.223.16.209	192.168.1.157	TLSv1…	117	Change Cipher Spec, Encrypted Handshake Message
627	2019-03-16 21:05:30.332833	192.168.1.157	17.248.135.92	TCP	78	59017 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=128 TSval=477686826 TSecr=0 SACK_PERM=1
628	2019-03-16 21:05:30.335363	192.168.1.157	192.168.1.1	DNS	80	Standard query 0x5810 A gsp-ssl.ls.apple.com
629	2019-03-16 21:05:30.336023	192.168.1.157	173.223.16.209	TCP	66	59016 → 443 [ACK] Seq=381 Ack=7272 Win=130944 Len=0 TSval=477686581 TSecr=692831814
630	2019-03-16 21:05:30.336641	17.248.135.92	192.168.1.157	TCP	74	443 → 59017 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=1921823275 TSecr=4
631	2019-03-16 21:05:30.347567	192.168.1.157	17.248.135.92	TCP	66	59017 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=477686839 TSecr=1921823275
632	2019-03-16 21:05:30.350584	192.168.1.1	192.168.1.157	DNS	313	Standard query response 0x5810 A gsp-ssl.ls.apple.com CNAME gsp-ssl.ls-apple.com.akadns.net CNAM
633	2019-03-16 21:05:30.362811	192.168.1.157	17.248.135.92	TLSv1…	321	Client Hello
634	2019-03-16 21:05:30.366681	17.248.135.92	192.168.1.157	TCP	66	443 → 59017 [ACK] Seq=1 Ack=256 Win=30048 Len=0 TSval=1921823305 TSecr=477686856
635	2019-03-16 21:05:30.369122	17.248.135.92	192.168.1.157	TLSv1…	1514	Server Hello
636	2019-03-16 21:05:30.369154	17.248.135.92	192.168.1.157	TCP	1514	443 → 59017 [ACK] Seq=1449 Ack=256 Win=30048 Len=1448 TSval=1921823308 TSecr=477686856 [TCP segm
637	2019-03-16 21:05:30.369338	192.168.1.157	173.223.16.209	TLSv1…	492	Application Data
638	2019-03-16 21:05:30.370170	17.248.135.92	192.168.1.157	TCP	1514	443 → 59017 [ACK] Seq=2897 Ack=256 Win=30048 Len=1448 TSval=1921823309 TSecr=477686856 [TCP segm
639	2019-03-16 21:05:30.370202	17.248.135.92	192.168.1.157	TLSv1…	1514	Certificate [TCP segment of a reassembled PDU]
640	2019-03-16 21:05:30.371208	17.248.135.92	192.168.1.157	TCP	1514	443 → 59017 [ACK] Seq=5793 Ack=256 Win=30048 Len=1448 TSval=1921823310 TSecr=477686856 [TCP segm
641	2019-03-16 21:05:30.371240	17.248.135.92	192.168.1.157	TLSv1…	1514	Certificate Status [TCP segment of a reassembled PDU]
642	2019-03-16 21:05:30.371896	192.168.1.157	17.248.135.92	TCP	66	59017 → 443 [ACK] Seq=256 Ack=2897 Win=128768 Len=0 TSval=477686864 TSecr=1921823308
643	2019-03-16 21:05:30.371921	192.168.1.157	17.248.135.92	TCP	66	59017 → 443 [ACK] Seq=256 Ack=5793 Win=125952 Len=0 TSval=477686864 TSecr=1921823309
644	2019-03-16 21:05:30.371940	192.168.1.157	17.248.135.92	TCP	66	[TCP Window Update] 59017 → 443 [ACK] Seq=256 Ack=5793 Win=131072 Len=0 TSval=477686864 TSecr=19
645	2019-03-16 21:05:30.371991	17.248.135.92	192.168.1.157	TLSv1…	337	Server Key Exchange, Server Hello Done

Packet log showing a first device, at 192.168.1.157, performs a DNS lookup to find an IP address associated with iCloud, which is 17.248.135.92, and initiates a TLS connection with a server, captured March 16, 2019.

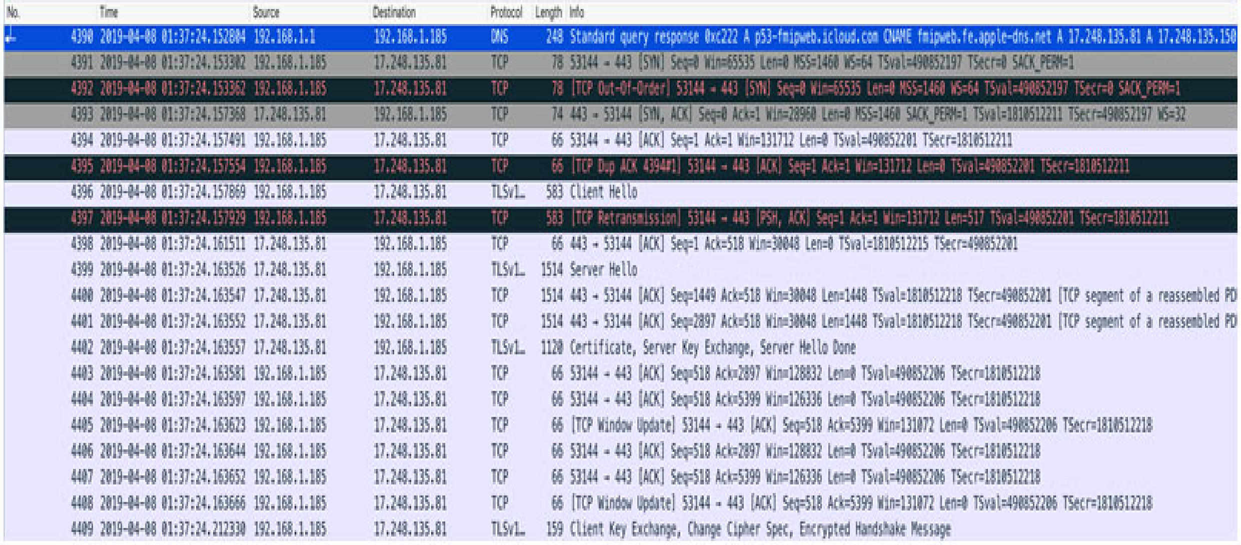






```
▾ General
    Request URL: https://p53-fmipweb.icloud.com/fmipservice/client/web/initClient?clientBuildNumber=1905Project37&clientId=CAB666CB-5C6F-427F-AED6-12F5FE
    92C2E7&clientMasteringNumber=1905B30&dsid=89683430
    Request Method: POST
    Status Code: ● 200 OK
    Remote Address: 17.248.187.199:443
    Referrer Policy: no-referrer-when-downgrade
▾ Response Headers
    Access-Control-Allow-Credentials: true
    Access-Control-Allow-Origin: https://www.icloud.com
    access-control-expose-headers: X-Apple-Request-UUID, Via
    Connection: keep-alive
```

Network log in Google Chrome web browser, captured April 8, 2018.



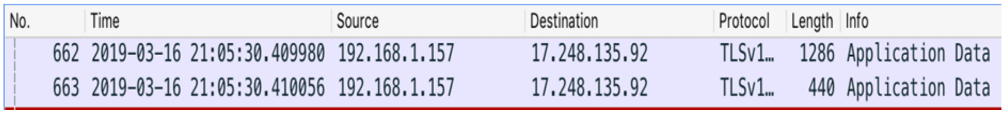
```
No.   Time                        Source         Destination    Protocol Length Info
4390  2019-04-08 01:37:24.152804  192.168.1.1    192.168.1.185  DNS      248 Standard query response 0xc222 A p53-fmipweb.icloud.com CNAME fmipweb.fe.apple-dns.net A 17.248.135.81 A 17.248.135.150
4391  2019-04-08 01:37:24.153302  192.168.1.185  17.248.135.81  TCP       78 53144 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=490852197 TSecr=0 SACK_PERM=1
4392  2019-04-08 01:37:24.153362  192.168.1.185  17.248.135.81  TCP       78 [TCP Out-Of-Order] 53144 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=490852197 TSecr=0 SACK_PERM=1
4393  2019-04-08 01:37:24.157368  17.248.135.81  192.168.1.185  TCP       74 443 → 53144 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=1810512211 TSecr=490852197 WS=32
4394  2019-04-08 01:37:24.157491  192.168.1.185  17.248.135.81  TCP       66 53144 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=490852201 TSecr=1810512211
4395  2019-04-08 01:37:24.157554  192.168.1.185  17.248.135.81  TCP       66 [TCP Dup ACK 4394#1] 53144 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=490852201 TSecr=1810512211
4396  2019-04-08 01:37:24.157869  192.168.1.185  17.248.135.81  TLSv1…   583 Client Hello
4397  2019-04-08 01:37:24.157929  192.168.1.185  17.248.135.81  TCP      583 [TCP Retransmission] 53144 → 443 [PSH, ACK] Seq=1 Ack=1 Win=131712 Len=517 TSval=490852201 TSecr=1810512211
4398  2019-04-08 01:37:24.161511  17.248.135.81  192.168.1.185  TCP       66 443 → 53144 [ACK] Seq=1 Ack=518 Win=30048 Len=0 TSval=1810512215 TSecr=490852201
4399  2019-04-08 01:37:24.163526  17.248.135.81  192.168.1.185  TLSv1…  1514 Server Hello
4400  2019-04-08 01:37:24.163547  17.248.135.81  192.168.1.185  TCP     1514 443 → 53144 [ACK] Seq=1449 Ack=518 Win=30048 Len=1448 TSval=1810512218 TSecr=490852201 [TCP segment of a reassembled PD
4401  2019-04-08 01:37:24.163552  17.248.135.81  192.168.1.185  TCP     1514 443 → 53144 [ACK] Seq=2897 Ack=518 Win=30048 Len=1448 TSval=1810512218 TSecr=490852201 [TCP segment of a reassembled PD
4402  2019-04-08 01:37:24.163557  17.248.135.81  192.168.1.185  TLSv1…  1120 Certificate, Server Key Exchange, Server Hello Done
4403  2019-04-08 01:37:24.163581  192.168.1.185  17.248.135.81  TCP       66 53144 → 443 [ACK] Seq=518 Ack=2897 Win=128832 Len=0 TSval=490852206 TSecr=1810512218
4404  2019-04-08 01:37:24.163597  192.168.1.185  17.248.135.81  TCP       66 53144 → 443 [ACK] Seq=518 Ack=5399 Win=126336 Len=0 TSval=490852206 TSecr=1810512218
4405  2019-04-08 01:37:24.163623  192.168.1.185  17.248.135.81  TCP       66 [TCP Window Update] 53144 → 443 [ACK] Seq=518 Ack=5399 Win=131072 Len=0 TSval=490852206 TSecr=1810512218
4406  2019-04-08 01:37:24.163644  192.168.1.185  17.248.135.81  TCP       66 53144 → 443 [ACK] Seq=518 Ack=2897 Win=128832 Len=0 TSval=490852206 TSecr=1810512218
4407  2019-04-08 01:37:24.163652  192.168.1.185  17.248.135.81  TCP       66 53144 → 443 [ACK] Seq=518 Ack=5399 Win=126336 Len=0 TSval=490852206 TSecr=1810512218
4408  2019-04-08 01:37:24.163666  192.168.1.185  17.248.135.81  TCP       66 [TCP Window Update] 53144 → 443 [ACK] Seq=518 Ack=5399 Win=131072 Len=0 TSval=490852206 TSecr=1810512218
4409  2019-04-08 01:37:24.212330  192.168.1.185  17.248.135.81  TLSv1…   159 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
```

Log showing that the web browser, at 192.168.1.185, performs a DNS lookup to find an IP address of associated with iCloud, which is 17.248.135.81, and initiates a TLS connection, captured April 8, 2019.

259. Claim 1 of the ’534 patent recites “receive a first message from the first device over the first connection.” The ’534 accused products satisfy this limitation. For example, the ’534 accused products receive a first message from the first device over the first connection, as confirmed through the exemplary documentation below.

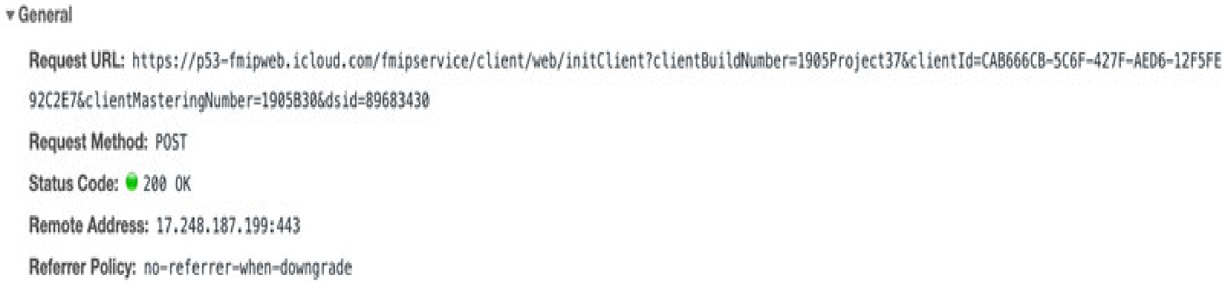
```
default  21:05:30.514154 -0400      FindMyFriends     [C1     Hostname#78e5c4e8:443     tcp,     url:     https://p28-fmfmobile.icloud.com/fmipservice/friends/1538423246/9387152d917ee7cc8ce9b3590b3cbe046fdad67d/first/initClien,     tls] start
```

iOS log from first iPhone launching Find My Friends Application, captured March 16, 2019.



No.	Time	Source	Destination	Protocol	Length	Info
662	2019-03-16 21:05:30.409980	192.168.1.157	17.248.135.92	TLSv1…	1286	Application Data
663	2019-03-16 21:05:30.410056	192.168.1.157	17.248.135.92	TLSv1…	440	Application Data

Log showing that a first device, at 192.168.1.157, sends application data via TLS to iCloud, which is at 17.248.135.92, captured March 16, 2019.



▾ General

Request URL: https://p53-fmipweb.icloud.com/fmipservice/client/web/initClient?clientBuildNumber=1905Project37&clientId=CAB666CB-5C6F-427F-AED6-12F5FE92C2E7&clientMasteringNumber=1905B30&dsid=89683430

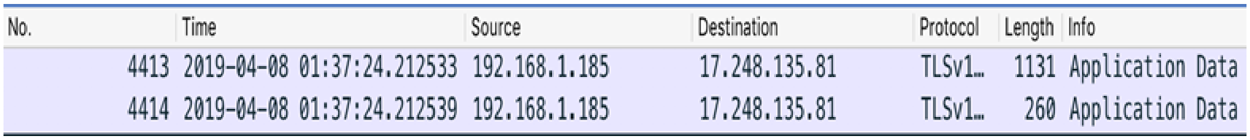
Request Method: POST

Status Code: ● 200 OK

Remote Address: 17.248.187.199:443

Referrer Policy: no-referrer-when-downgrade

Google Chrome network log when launching Find My iPhone, captured April 8, 2019.



No.	Time	Source	Destination	Protocol	Length	Info
4413	2019-04-08 01:37:24.212533	192.168.1.185	17.248.135.81	TLSv1…	1131	Application Data
4414	2019-04-08 01:37:24.212539	192.168.1.185	17.248.135.81	TLSv1…	260	Application Data

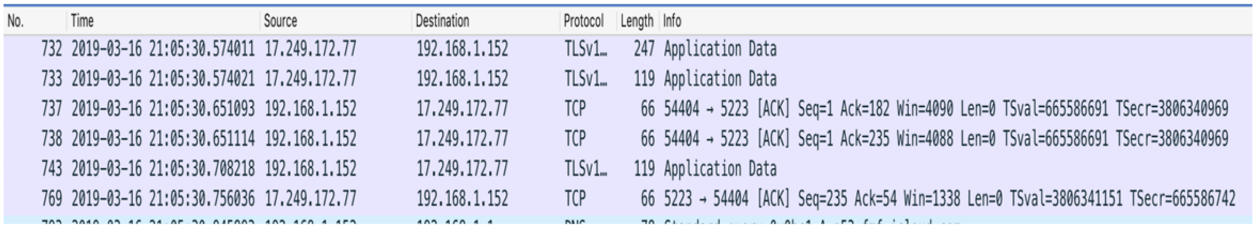
Packet log showing that the web browser, at 192.168.1.185, sends application data via TLS to iCloud, which is at 17.248.135.81, captured April 8, 2019.

260. Claim 1 of the ’534 patent recites “generate a second message for a second device based on the first message from the first device; send the second message to the second device.” The ’534 accused products satisfy this limitation. For example, the ’534 accused products use the Apple Push Notification Service (APNs) to send a message and instruct the

second device to send a message (e.g., an updated location), as confirmed through the exemplary documentation below.

```
default 21:05:30.662300 -0400     apsd     2019-03-16  21:05:30  -0400  apsd[85]:  <APSCourier:  0x11bd0d530>:
Outstanding data received: <0a7d8101 42147f29 2acab37b ed771b36 b09e612e 395eee13 a69a0480 05a80106 a7158c99 7c44677e
47078003 3f127b22 73657276 6572436f 6e746578 74223a7b 22746170 53656e64 5453223a 22323031 392d3033 2d313754 30313a30
353a3330 2e353331 5a222c22 74617053 656e6443 6f6e7465 7874223a 22666d66 227d7d0d 020181> (length 131) onInterface:
NonCellular. Connected on 1 interfaces.

default 21:05:30.665491 -0400     apsd     2019-03-16 21:05:30 -0400 apsd[85]: <APSCourier: 0x11bd0d530>: Stream
processing: complete yes, invalid no, length parsed 127, parameters {

APSProtocolCommand = 10;

APSProtocolMessageExpiry = "1970-01-01 18:12:15 +0000";

APSProtocolMessageID = <00000000>;

APSProtocolMessageTimestamp = 1552784730538606151;

APSProtocolPayload = <7b227365 72766572 436f6e74 65787422 3a7b2274 61705365 6e645453 223a2232 3031392d 30332d31
37543031 3a30353a 33302e35 33315a22 2c227461 7053656e 64436f6e 74657874 223a2266 6d66227d 7d>;

APSProtocolToken = <6dc220e2 a40b027a d01c77ce 7df3f958 fdbb9ec5 bf218587 fc444b14 e2e9c4b1>;

APSProtocolTopicHash = <7f292aca b37bed77 1b36b09e 612e395e ee13a69a>;

}

default 21:05:30.667717 -0400     apsd     2019-03-16 21:05:30 -0400 apsd[85]: copyTokenForDomain push.apple.com
(null)

default 21:05:30.681194 -0400     apsd     2019-03-16 21:05:30 -0400 apsd[85]: copyTokenForDomain push.apple.com
(null)

default 21:05:30.693112 -0400     apsd     2019-03-16 21:05:30 -0400 apsd[85]: copyTokenForDomain push.apple.com
(null)

default 21:05:30.700560 -0400     apsd     2019-03-16 21:05:30 -0400 apsd[85]: copyTokenForDomain push.apple.com
(null)

default 21:05:30.706677 -0400     apsd     2019-03-16  21:05:30  -0400  apsd[85]:  <APSPushHistory:  0x11bd405d0>
timestampForTopic?  com.apple.mobileme.fmf  token  <6dc220e2  a40b027a  d01c77ce  7df3f958  fdbb9ec5  bf218587  fc444b14
e2e9c4b1>

default 21:05:30.707061 -0400     apsd     2019-03-16 21:05:30 -0400 apsd[85]: copyTokenForDomain push.apple.com
,7f292affffffcaffffffb37bffffffed771b36ffffffb0ffffff9e612e395effffffee13ffffffa6ffffff9a

default 21:05:30.713395 -0400     apsd     2019-03-16  21:05:30  -0400  apsd[85]:  <APSPushHistory:  0x11bd405d0>
timestampForTopic? com.apple.mobileme.fmf token (null)

default 21:05:30.714018 -0400     apsd     2019-03-16  21:05:30  -0400  apsd[85]:  <APSPushHistory:  0x11bd405d0>
receivedPushWithTopic  com.apple.mobileme.fmf  token  <6dc220e2  a40b027a  d01c77ce  7df3f958  fdbb9ec5  bf218587  fc444b14
e2e9c4b1> payload <7b227365 72766572 436f6e74 65787422 3a7b2274 61705365 6e645453 223a2232 3031392d 30332d31 37543031
3a30353a 33302e35 33315a22 2c227461 7053656e 64436f6e 74657874 223a2266 6d66227d 7d> timestamp 2019-03-17 01:05:30
+0000
```

Log from a second device showing that apsd – the APNs client on the second device – receives a push notification with topic “com.apple.mobileme.fmf,” captured March 16, 2019.



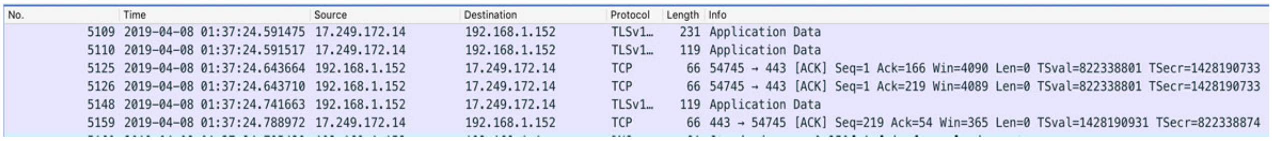
No.	Time	Source	Destination	Protocol	Length	Info
732	2019-03-16 21:05:30.574011	17.249.172.77	192.168.1.152	TLSv1…	247	Application Data
733	2019-03-16 21:05:30.574021	17.249.172.77	192.168.1.152	TLSv1…	119	Application Data
737	2019-03-16 21:05:30.651093	192.168.1.152	17.249.172.77	TCP	66	54404 → 5223 [ACK] Seq=1 Ack=182 Win=4090 Len=0 TSval=665586691 TSecr=3806340969
738	2019-03-16 21:05:30.651114	192.168.1.152	17.249.172.77	TCP	66	54404 → 5223 [ACK] Seq=1 Ack=235 Win=4088 Len=0 TSval=665586691 TSecr=3806340969
743	2019-03-16 21:05:30.708218	192.168.1.152	17.249.172.77	TLSv1…	119	Application Data
769	2019-03-16 21:05:30.756036	17.249.172.77	192.168.1.152	TCP	66	5223 → 54404 [ACK] Seq=235 Ack=54 Win=1338 Len=0 TSval=3806341151 TSecr=665586742

Packet log showing transmission of push notification from APNs at 17.249.172.77

```
default 01:37:24.649886 -0400 apsd 2019-04-08 01:37:24 -0400 apsd[87]: <APSCourier: 0x10021a020>:
Outstanding data received: <0a6a8101 82000480 05a80106 a7159369 1728eff2 46078003 3f137b22 73657276 6572436f 6e746578
74223a7b 22746170 53656e64 5453223a 22323031 392d3034 2d303854 30353a33 373a3234 2e353035 5a222c22 74617053 656e6443
6f6e7465 7874223a 22666d69 70227d7d 0d020181> (length 112) onInterface: NonCellular. Connected on 1 interfaces.

default 01:37:24.651626 -0400 apsd 2019-04-08 01:37:24 -0400 apsd[87]: <APSCourier: 0x10021a020>: Stream
processing: complete yes, invalid no, length parsed 108, parameters {

    APSProtocolCommand = 10;

    APSProtocolMessageExpiry = "1970-01-01 18:12:15 +0000";

    APSProtocolMessageID = <00000000>;

    APSProtocolMessageTimestamp = 1554701844564931142;

    APSProtocolPayload = <7b227365 72766572 436f6e74 65787422 3a7b2274 61705365 6e645453 223a2232 3031392d 30342d30
38543035 3a33373a 32342e35 30355a22 2c227461 7053656e 64436f6e 74657874 223a2266 6d697022 7d7d>;

    APSProtocolToken = <6dc220e2 a40b027a d01c77ce 7df3f958 fdbb9ec5 bf218587 fc444b14 e2e9c4b1>;

    APSProtocolTopicHash = <79afbbad c8f8142d 144202ed 12106d5c d3f88f1a>;

}

default 01:37:24.653420 -0400 apsd 2019-04-08 01:37:24 -0400 apsd[87]: copyTokenForDomain push.apple.com
(null)

default 01:37:24.670837 -0400 apsd 2019-04-08 01:37:24 -0400 apsd[87]: copyTokenForDomain push.apple.com
(null)

default 01:37:24.690991 -0400 apsd 2019-04-08 01:37:24 -0400 apsd[87]: copyTokenForDomain push.apple.com
(null)

default 01:37:24.710164 -0400 apsd 2019-04-08 01:37:24 -0400 apsd[87]: copyTokenForDomain push.apple.com
(null)

default 01:37:24.722474 -0400 apsd 2019-04-08 01:37:24 -0400 apsd[87]: <APSPushHistory: 0x10031cba0>
timestampForTopic? com.apple.mobileme.fmip token <6dc220e2 a40b027a d01c77ce 7df3f958 fdbb9ec5 bf218587 fc444b14
e2e9c4b1>

default 01:37:24.722959 -0400 apsd 2019-04-08 01:37:24 -0400 apsd[87]: copyTokenForDomain push.apple.com
,79ffffffaffffffbbffffffadffffffc8fffffff8142d144202ffffffed12106d5cffffffd3fffffff8ffffff8f1a

default 01:37:24.731136 -0400 apsd 2019-04-08 01:37:24 -0400 apsd[87]: <APSPushHistory: 0x10031cba0>
timestampForTopic? com.apple.mobileme.fmip token (null)

default 01:37:24.731660 -0400 apsd 2019-04-08 01:37:24 -0400 apsd[87]: <APSPushHistory: 0x10031cba0>
receivedPushWithTopic com.apple.mobileme.fmip token <6dc220e2 a40b027a d01c77ce 7df3f958 fdbb9ec5 bf218587 fc444b14
e2e9c4b1> payload <7b227365 72766572 436f6e74 65787422 3a7b2274 61705365 6e645453 223a2232 3031392d 30342d30 38543035
3a33373a 32342e35 30355a22 2c227461 7053656e 64436f6e 74657874 223a2266 6d697022 7d7d> timestamp 2019-04-08 05:37:24
+0000

default 01:37:24.732110 -0400 apsd 2019-04-08 01:37:24 -0400 apsd[87]: <APSCourier: 0x10021a020>: Received
message for enabled topic 'com.apple.mobileme.fmip' with payload '{

    serverContext =     {
        tapSendContext = fmip;

        tapSendTS = "2019-04-08T05:37:24.505Z";

    };

}' onInterface: NonCellular  for device token: YES  with priority (null)

default  01:37:24.732506 -0400      apsd     2019-04-08 01:37:24 -0400 apsd[87]: <APSMessageStore: 0x1002baef0> asked
to  store  incoming  message  <APSIncomingMessage:  0x1002dcc20>  with  guid  6C08561C-CAA0-44DD-8730-257D976F6154
environment <APSEnvironment: 0x1002193f0: production>

default  01:37:24.733159 -0400      apsd     2019-04-08 01:37:24 -0400 apsd[87]: <APSCourier: 0x10021a020>: Calling
into AWD for PushReceived

default  01:37:24.733691 -0400      apsd     2019-04-08 01:37:24 -0400 apsd[87]: <APSCourier: 0x10021a020>: AWD for
PushReceived finished

default  01:37:24.734181 -0400      apsd     2019-04-08 01:37:24 -0400 apsd[87]: <APSCourier: 0x10021a020>: Stream
processing: complete yes, invalid no, length parsed 4, parameters {

    APSProtocolCommand = 13;

    APSProtocolKeepAliveResponse = 1;

}

default  01:37:24.734840 -0400      apsd     2019-04-08 01:37:24 -0400 apsd[87]: <APSCourier: 0x10021a020>: Received
keep-alive response 1 on interface NonCellular: {

    APSProtocolCommand = 13;

    APSProtocolKeepAliveResponse = 1;

}

default  01:37:24.735170 -0400      apsd     2019-04-08 01:37:24 -0400 apsd[87]: APSMessageStore - New message record
[<APSIncomingMessageRecord 0x1003f2ca0 [0x1b24f8bb8]>{}] has ID [41321].

default  01:37:24.735846 -0400      apsd     2019-04-08 01:37:24 -0400 apsd[87]: <APSCourier: 0x10021a020>: Stream
processing: complete no, invalid no, length parsed 0, parameters (null)

default  01:37:24.736592 -0400      apsd     2019-04-08 01:37:24 -0400 apsd[87]: <APSMessageStore: 0x1002baef0>
calling completion block for incoming message 6C08561C-CAA0-44DD-8730-257D976F6154

default  01:37:24.737586 -0400      apsd     2019-04-08 01:37:24 -0400 apsd[87]: <APSCourier: 0x10021a020>: Sending
acknowledgement message with response 0 and messageId <00000000> (0)

default  01:37:24.737997 -0400      apsd     2019-04-08 01:37:24 -0400 apsd[87]: <APSCourier: 0x10021a020> Noting
push - using last lq 0 and rat (null)  (instead of -2, kCTRegistrationRadioAccessTechnologyUnknown)

default  01:37:24.738740 -0400      apsd     2019-04-08      01:37:24      -0400      apsd[87]:      <APSDecayTimer:
0x10021aef0>:APSNetworkMonitor decaying cost (885 - 14) = 871 for 51.098976 seconds

default  01:37:24.739619 -0400      apsd     2019-04-08      01:37:24      -0400      apsd[87]:      <APSDecayTimer:
0x10021aef0>:APSNetworkMonitor addCost: 166 - _currentCost is now 1037

default  01:37:24.740403 -0400      apsd     2019-04-08   01:37:24   -0400   apsd[87]:   <APSCourier:   0x10021a020>
_notifyForIncomingMessage <APSIncomingMessage: 0x1002dcc20> with guid 6C08561C-CAA0-44DD-8730-257D976F6154

default  01:37:24.742232 -0400      apsd     2019-04-08 01:37:24 -0400 apsd[87]: Dispatching high priority message:
<OS_xpc_dictionary: <dictionary: 0x1003ebe70> { count = 2, transaction: 0, voucher = 0x0, contents =

         "message" => <dictionary: 0x10321a730> { count = 1, transaction: 0, voucher = 0x0, contents =

                  "ECF19A18-7AA6-4141-B4DC-A2E5123B2B5C" => <data: 0x1003bf8e0>: { length = 16384 bytes, contents =
0x62706c697374313513a1010000000000801200000000d87f... }

         }

         "message-type" => <int64: 0x1003c3e60>: 30

}>
```

```
default  01:37:24.749622 -0400      findmydeviced     Push message received <private>
```

Log from second device showing that apsd – the APNs client on the second device – receives a push notification with topic “com.apple.mobileme.fmip” and forwards it to the “findmydeviced” daemon running on the second device, captured April 8, 2019.



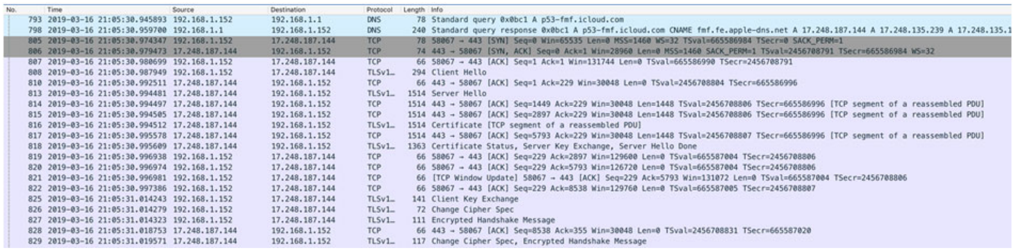
```
No.    Time                        Source          Destination     Protocol  Length  Info
5109   2019-04-08 01:37:24.591475  17.249.172.14   192.168.1.152   TLSv1…     231   Application Data
5110   2019-04-08 01:37:24.591517  17.249.172.14   192.168.1.152   TLSv1…     119   Application Data
5125   2019-04-08 01:37:24.643664  192.168.1.152   17.249.172.14   TCP         66   54745 → 443 [ACK] Seq=1 Ack=166 Win=4090 Len=0 TSval=822338801 TSecr=1428190733
5126   2019-04-08 01:37:24.643710  192.168.1.152   17.249.172.14   TCP         66   54745 → 443 [ACK] Seq=1 Ack=219 Win=4089 Len=0 TSval=822338801 TSecr=1428190733
5148   2019-04-08 01:37:24.741663  192.168.1.152   17.249.172.14   TLSv1…     119   Application Data
5159   2019-04-08 01:37:24.788972  17.249.172.14   192.168.1.152   TCP         66   443 → 54745 [ACK] Seq=219 Ack=54 Win=365 Len=0 TSval=1428190931 TSecr=822338874
```

Log showing transmission of push notification from APNs at 17.249.172.14 to the second device, 192.168.1.152, captured April 8, 2019.

261. Claim 1 of the ’534 patent recites “receive a second connection associated with the second device, wherein the first connection includes a connection that is initiated by the first device, wherein the second connection includes a connection that is initiated by the second device.” The ’534 accused products satisfy this limitation. For example, upon receiving the second message over APNs, the second device (using for example “fmflocatord”) connects to iCloud, as confirmed by the exemplary documentation below.

```
default  21:05:30.956905 -0400      fmflocatord       TIC TCP Conn Start [1:0x100251620]
default  21:05:30.993832 -0400      fmflocatord       TIC TCP Conn Event [1:0x100251620]: 1 Err(0)
default  21:05:30.993907 -0400      fmflocatord       TIC TCP Conn Connected [1:0x100251620]: Err(0)
default  21:05:30.994148 -0400      fmflocatord       TIC Enabling TLS [1:0x100251620]
default  21:05:31.004275 -0400      fmflocatord       TIC TLS Event [1:0x100251620]: 2, Pending(0)
default  21:05:31.005090 -0400      fmflocatord       TIC TLS Event [1:0x100251620]: 11, Pending(0)
default  21:05:31.005919 -0400      fmflocatord       TIC TLS Event [1:0x100251620]: 22, Pending(0)
default  21:05:31.006194 -0400      fmflocatord       TIC TLS Event [1:0x100251620]: 12, Pending(0)
default  21:05:31.006295 -0400      fmflocatord       TIC TLS Event [1:0x100251620]: 14, Pending(0)
```

iOS log from a second device, captured March 16, 2019.

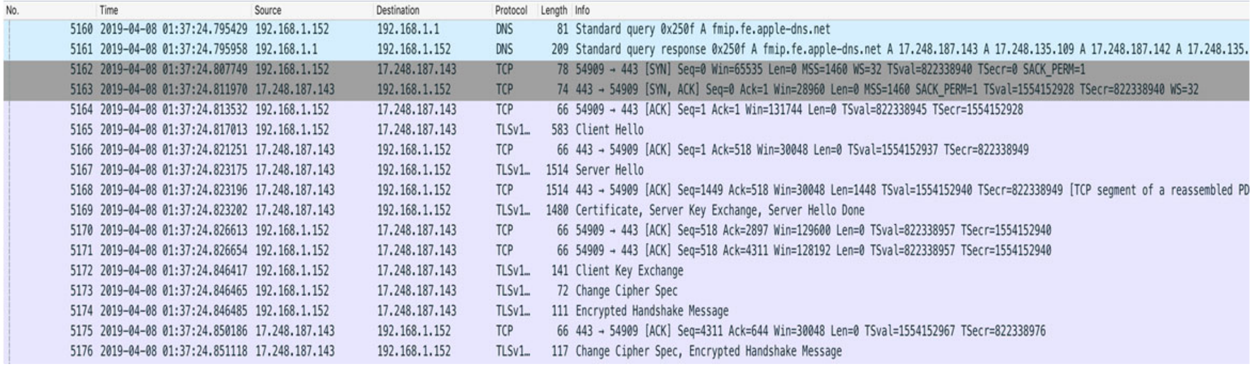


```
No.  Time                        Source          Destination     Protocol  Length Info
793  2019-03-16 21:05:30.945893  192.168.1.152   192.168.1.1     DNS         78  Standard query 0x0bc1 A p53-fmf.icloud.com
798  2019-03-16 21:05:30.959700  192.168.1.1     192.168.1.152   DNS        240  Standard query response 0x0bc1 A p53-fmf.icloud.com CNAME fmf.fe.apple-dns.net A 17.248.187.144 A 17.248.135.239 A 17.248.135.1
805  2019-03-16 21:05:30.974347  192.168.1.152   17.248.187.144  TCP         78  58067 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=665586984 TSecr=0 SACK_PERM=1
806  2019-03-16 21:05:30.979473  17.248.187.144  192.168.1.152   TCP         74  443 → 58067 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=2456708791 TSecr=665586984 WS=32
807  2019-03-16 21:05:30.980699  192.168.1.152   17.248.187.144  TCP         66  58067 → 443 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=665586990 TSecr=2456708791
808  2019-03-16 21:05:30.987949  192.168.1.152   17.248.187.144  TLSv1…     294  Client Hello
810  2019-03-16 21:05:30.992511  17.248.187.144  192.168.1.152   TCP         66  443 → 58067 [ACK] Seq=1 Ack=229 Win=30048 Len=0 TSval=2456708804 TSecr=665586996
813  2019-03-16 21:05:30.994481  17.248.187.144  192.168.1.152   TLSv1…    1514  Server Hello
814  2019-03-16 21:05:30.994497  17.248.187.144  192.168.1.152   TCP       1514  443 → 58067 [ACK] Seq=1449 Ack=229 Win=30048 Len=1448 TSval=2456708806 TSecr=665586996 [TCP segment of a reassembled PDU]
815  2019-03-16 21:05:30.994505  17.248.187.144  192.168.1.152   TCP       1514  443 → 58067 [ACK] Seq=2897 Ack=229 Win=30048 Len=1448 TSval=2456708806 TSecr=665586996 [TCP segment of a reassembled PDU]
816  2019-03-16 21:05:30.994512  17.248.187.144  192.168.1.152   TLSv1…    1514  Certificate [TCP segment of a reassembled PDU]
817  2019-03-16 21:05:30.995578  17.248.187.144  192.168.1.152   TCP       1514  443 → 58067 [ACK] Seq=5793 Ack=229 Win=30048 Len=1448 TSval=2456708807 TSecr=665586996 [TCP segment of a reassembled PDU]
818  2019-03-16 21:05:30.995609  17.248.187.144  192.168.1.152   TLSv1…    1363  Certificate Status, Server Key Exchange, Server Hello Done
819  2019-03-16 21:05:30.996938  192.168.1.152   17.248.187.144  TCP         66  58067 → 443 [ACK] Seq=229 Ack=2897 Win=129600 Len=0 TSval=665587004 TSecr=2456708806
820  2019-03-16 21:05:30.996974  192.168.1.152   17.248.187.144  TCP         66  58067 → 443 [ACK] Seq=229 Ack=5793 Win=126720 Len=0 TSval=665587004 TSecr=2456708806
821  2019-03-16 21:05:30.996981  192.168.1.152   17.248.187.144  TCP         66  [TCP Window Update] 58067 → 443 [ACK] Seq=229 Ack=5793 Win=131072 Len=0 TSval=665587004 TSecr=2456708806
822  2019-03-16 21:05:30.997386  192.168.1.152   17.248.187.144  TCP         66  58067 → 443 [ACK] Seq=229 Ack=8538 Win=129760 Len=0 TSval=665587005 TSecr=2456708807
825  2019-03-16 21:05:31.014243  192.168.1.152   17.248.187.144  TLSv1…     141  Client Key Exchange
826  2019-03-16 21:05:31.014279  192.168.1.152   17.248.187.144  TLSv1…      72  Change Cipher Spec
827  2019-03-16 21:05:31.014323  192.168.1.152   17.248.187.144  TLSv1…     111  Encrypted Handshake Message
828  2019-03-16 21:05:31.018753  17.248.187.144  192.168.1.152   TCP         66  443 → 58067 [ACK] Seq=8538 Ack=355 Win=30048 Len=0 TSval=2456708831 TSecr=665587020
829  2019-03-16 21:05:31.019571  17.248.187.144  192.168.1.152   TLSv1…     117  Change Cipher Spec, Encrypted Handshake Message
```

Packet log showing the second device, at 192.168.1.152, initiates a TLS connection to iCloud, 17.248.187.144, captured March 16, 2019.

```
default  01:37:24.784789 -0400      findmydeviced     <private> Sending to url <private>
default  01:37:24.785303 -0400      findmydeviced     <private> Sending headers:
<private>
default  01:37:24.786185 -0400      findmydeviced     <private> Sending with body dict :
<private>
default  01:37:24.797619 -0400      findmydeviced     TIC TCP Conn Start [12:0x100553050]
default  01:37:24.867680 -0400      findmydeviced     TIC TCP Conn Event [12:0x100553050]: 1 Err(0)
default  01:37:24.867767 -0400      findmydeviced     TIC TCP Conn Connected [12:0x100553050]: Err(0)
default  01:37:24.867870 -0400      findmydeviced     TIC Enabling TLS [12:0x100553050]
default  01:37:24.869276 -0400      findmydeviced     TIC TLS Event [12:0x100553050]: 2, Pending(0)
default  01:37:24.869316 -0400      findmydeviced     TIC TLS Event [12:0x100553050]: 11, Pending(0)
default  01:37:24.869447 -0400      findmydeviced     TIC TLS Event [12:0x100553050]: 12, Pending(0)
default  01:37:24.869527 -0400      findmydeviced     TIC TLS Event [12:0x100553050]: 14, Pending(0)
default  01:37:24.869694 -0400      findmydeviced     could not disable pinning: not an internal release
default  01:37:24.869818 -0400      findmydeviced     TIC TLS Trust Result [12:0x100553050]: 0
default  01:37:24.871668 -0400      findmydeviced     TIC TCP Conn Event [12:0x100553050]: 8 Err(0)
default  01:37:24.871768 -0400      findmydeviced     TIC TLS Handshake Complete [12:0x100553050]
```

Log from second device showing connection to FMIP iCloud initiated by findmydeviced, captured April 8, 2019.



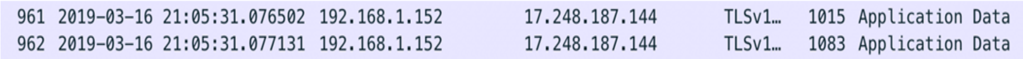
No.	Time	Source	Destination	Protocol	Length	Info
5160	2019-04-08 01:37:24.795429	192.168.1.152	192.168.1.1	DNS	81	Standard query 0x250f A fmip.fe.apple-dns.net
5161	2019-04-08 01:37:24.795958	192.168.1.1	192.168.1.152	DNS	209	Standard query response 0x250f A fmip.fe.apple-dns.net A 17.248.187.143 A 17.248.135.109 A 17.248.187.142 A 17.248.135.
5162	2019-04-08 01:37:24.807749	192.168.1.152	17.248.187.143	TCP	78	54909 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=822338940 TSecr=0 SACK_PERM=1
5163	2019-04-08 01:37:24.811970	17.248.187.143	192.168.1.152	TCP	74	443 → 54909 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=1554152928 TSecr=822338940 WS=32
5164	2019-04-08 01:37:24.813532	192.168.1.152	17.248.187.143	TCP	66	54909 → 443 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=822338945 TSecr=1554152928
5165	2019-04-08 01:37:24.817013	192.168.1.152	17.248.187.143	TLSv1…	583	Client Hello
5166	2019-04-08 01:37:24.821251	17.248.187.143	192.168.1.152	TCP	66	443 → 54909 [ACK] Seq=1 Ack=518 Win=30048 Len=0 TSval=1554152937 TSecr=822338949
5167	2019-04-08 01:37:24.823175	17.248.187.143	192.168.1.152	TLSv1…	1514	Server Hello
5168	2019-04-08 01:37:24.823196	17.248.187.143	192.168.1.152	TCP	1514	443 → 54909 [ACK] Seq=1449 Ack=518 Win=30048 Len=1448 TSval=1554152940 TSecr=822338949 [TCP segment of a reassembled PD
5169	2019-04-08 01:37:24.823202	17.248.187.143	192.168.1.152	TLSv1…	1480	Certificate, Server Key Exchange, Server Hello Done
5170	2019-04-08 01:37:24.826613	192.168.1.152	17.248.187.143	TCP	66	54909 → 443 [ACK] Seq=518 Ack=2897 Win=129600 Len=0 TSval=822338957 TSecr=1554152940
5171	2019-04-08 01:37:24.826654	192.168.1.152	17.248.187.143	TCP	66	54909 → 443 [ACK] Seq=518 Ack=4311 Win=128192 Len=0 TSval=822338957 TSecr=1554152940
5172	2019-04-08 01:37:24.846417	192.168.1.152	17.248.187.143	TLSv1…	141	Client Key Exchange
5173	2019-04-08 01:37:24.846465	192.168.1.152	17.248.187.143	TLSv1…	72	Change Cipher Spec
5174	2019-04-08 01:37:24.846485	192.168.1.152	17.248.187.143	TLSv1…	111	Encrypted Handshake Message
5175	2019-04-08 01:37:24.850186	17.248.187.143	192.168.1.152	TCP	66	443 → 54909 [ACK] Seq=4311 Ack=644 Win=30048 Len=0 TSval=1554152967 TSecr=822338976
5176	2019-04-08 01:37:24.851118	17.248.187.143	192.168.1.152	TLSv1…	117	Change Cipher Spec, Encrypted Handshake Message

Log showing that the second device, at 192.168.1.152, initiates a TLS connection to fmip.fe.apple-dns.net, at 17.248.187.143, captured April 8, 2019.

262. Claim 1 of the ’534 patent recites “receive a third message from the second device, wherein the third message is generated by the second device in response to receipt of the second message, wherein the third message contains a latest version of data stored at the second device and wherein the first message comprises a data query that is transmitted from the first device for the latest version of the data stored at the second device.” The ’534 accused products satisfy this limitation. For example, once the second device obtains the current location of the device, it sends that location to iCloud over a connection that the second device initiates.

```
default  21:05:31.387268 -0400      fmflocatord      delivering locations to client's delegate
default  21:05:31.388181 -0400      fmflocatord      delivering locations to client's delegate
default  21:05:31.388459 -0400      fmflocatord      delivering locations to client's delegate
default  21:05:31.405716 -0400      fmflocatord      TIC TCP Conn Start [3:0x100254c90]
default  21:05:31.581577 -0400      fmflocatord      TIC TCP Conn Event [3:0x100254c90]: 1 Err(0)
default  21:05:31.581643 -0400      fmflocatord      TIC TCP Conn Connected [3:0x100254c90]: Err(0)
default  21:05:31.581689 -0400      fmflocatord      delivering locations to client's delegate
default  21:05:31.582299 -0400      fmflocatord      TIC Enabling TLS [3:0x100254c90]
default  21:05:31.690216 -0400      fmflocatord      TIC TLS Event [3:0x100254c90]: 2, Pending(0)
default  21:05:31.690606 -0400      fmflocatord      TIC TLS Event [3:0x100254c90]: 20, Pending(0)
default  21:05:31.692790 -0400      fmflocatord      TIC TCP Conn Event [3:0x100254c90]: 8 Err(0)
default  21:05:31.692940 -0400      fmflocatord      TIC TLS Handshake Complete [3:0x100254c90]
```

iOS log from a second device showing a connection to iCloud is initiated by fmflocatord, captured March 16, 2019.



```
961 2019-03-16 21:05:31.076502 192.168.1.152       17.248.187.144      TLSv1…   1015 Application Data
962 2019-03-16 21:05:31.077131 192.168.1.152       17.248.187.144      TLSv1…   1083 Application Data
```

Packet log showing that the second device, at 192.168.1.152, sends application data to iCloud, at 17.248.187.144, over the TLS connection, captured March 16, 2019.

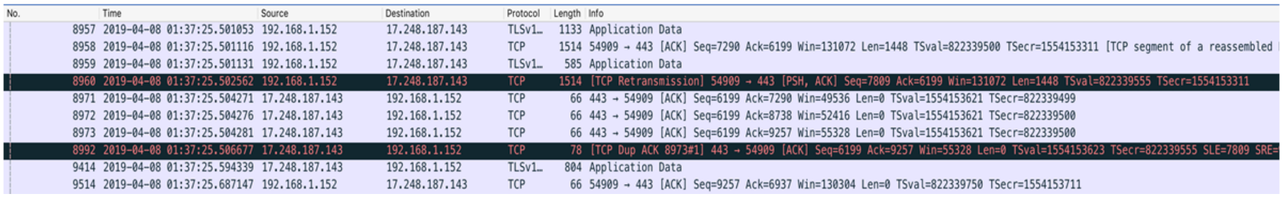
263. Claim 1 of the ’534 patent recites “generate a fourth message, wherein the fourth message includes data from the third message; and.” The ’534 accused products satisfy this limitation. For example, Apple’s server forwards the second device’s location back to the first device.

```
default  21:05:32.243607 -0400      FindMyFriends     Update from server: 2 locateInProgress changed
```

iOS log from first device showing receipt of updated location data by Find My Friends App.

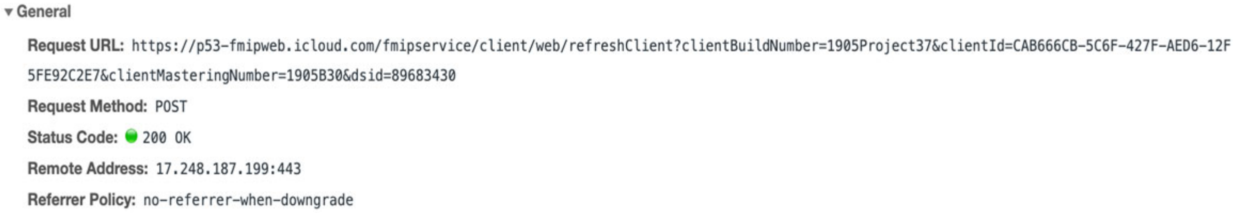
```
default  01:37:25.430133 -0400      findmydeviced     <private> Received location with Position Type = (4), Accuracy
= <private>, Latitude = <private>, Longitude = <private>, Timestamp = Mon Apr  8 01:37:25 2019
default  01:37:25.430269 -0400      findmydeviced     <private>  Location  has  accuracy  within  current  publish
threshold of 8000.00. Publishing it within the next 2 seconds
default  01:37:25.430323 -0400      findmydeviced     <private> New publish threshold is 59.57
default  01:37:25.430439 -0400      findmydeviced     <private>  Location  type  changed  from  1  to  4  with  distance
traveled 15.87. Publishing it immediately
default  01:37:25.430533 -0400      findmydeviced     <private> Publishing the location to the server for reason 4
default  01:37:25.430682 -0400      findmydeviced     Publishing  Location  with  Accuracy:  <private>  Longitude:
<private> Latitude: <private>
default  01:37:25.430759 -0400      findmydeviced     <private> Enqueueing request
default  01:37:25.430840 -0400      findmydeviced     <private> Sending to channel <private>
default  01:37:25.434241 -0400      findmydeviced     FMDAccessoryRegistry paired with 0 accessories
default  01:37:25.434719 -0400      findmydeviced     Reading  in  field  collection  status  :  secState  -  (null)
<private>
default  01:37:25.434821 -0400      findmydeviced     System group container path is <private>
default  01:37:25.436795 -0400      findmydeviced     There was a file read error while getting the protected context
serverContextHeaderContext.
default  01:37:25.437000 -0400      findmydeviced     <private> Sending to url <private>
default  01:37:25.437176 -0400      findmydeviced     <private> Sending headers:
<private>
default  01:37:25.437282 -0400      findmydeviced     <private> Sending with body dict :
<private>
```

Log from second device showing sending of the third location message once location is obtained, captured April 8, 2019.



```
No.   Time                       Source          Destination     Protocol Length Info
 8957 2019-04-08 01:37:25.501053 192.168.1.152   17.248.187.143  TLSv1…   1133 Application Data
 8958 2019-04-08 01:37:25.501116 192.168.1.152   17.248.187.143  TCP      1514 54909 → 443 [ACK] Seq=7290 Ack=6199 Win=131072 Len=1448 TSval=822339500 TSecr=1554153311 [TCP segment of a reassembled
 8959 2019-04-08 01:37:25.501131 192.168.1.152   17.248.187.143  TLSv1…    585 Application Data
 8960 2019-04-08 01:37:25.502562 192.168.1.152   17.248.187.143  TCP      1514 [TCP Retransmission] 54909 → 443 [PSH, ACK] Seq=7809 Ack=6199 Win=131072 Len=1448 TSval=822339555 TSecr=1554153311
 8971 2019-04-08 01:37:25.504271 17.248.187.143  192.168.1.152   TCP        66 443 → 54909 [ACK] Seq=6199 Ack=7290 Win=49536 Len=0 TSval=1554153621 TSecr=822339499
 8972 2019-04-08 01:37:25.504276 17.248.187.143  192.168.1.152   TCP        66 443 → 54909 [ACK] Seq=6199 Ack=8738 Win=52416 Len=0 TSval=1554153621 TSecr=822339500
 8973 2019-04-08 01:37:25.504281 17.248.187.143  192.168.1.152   TCP        66 443 → 54909 [ACK] Seq=6199 Ack=9257 Win=55328 Len=0 TSval=1554153621 TSecr=822339500
 8992 2019-04-08 01:37:25.506677 17.248.187.143  192.168.1.152   TCP        78 [TCP Dup ACK 8973#1] 443 → 54909 [ACK] Seq=6199 Ack=9257 Win=55328 Len=0 TSval=1554153623 TSecr=822339555 SLE=7809 SRE=
 9414 2019-04-08 01:37:25.594339 17.248.187.143  192.168.1.152   TLSv1…    804 Application Data
 9514 2019-04-08 01:37:25.687147 192.168.1.152   17.248.187.143  TCP        66 54909 → 443 [ACK] Seq=9257 Ack=6937 Win=130304 Len=0 TSval=822339750 TSecr=1554153711
```

Log showing that the second device, at 192.168.1.152, sends application data to fmip.fe.apple-dns.net, at 17.248.187.143, over the TLS connection, captured April 8, 2019.



▾ General

Request URL: https://p53-fmipweb.icloud.com/fmipservice/client/web/refreshClient?clientBuildNumber=1905Project37&clientId=CAB666CB-5C6F-427F-AED6-12F5FE92C2E7&clientMasteringNumber=1905B30&dsid=89683430

Request Method: POST

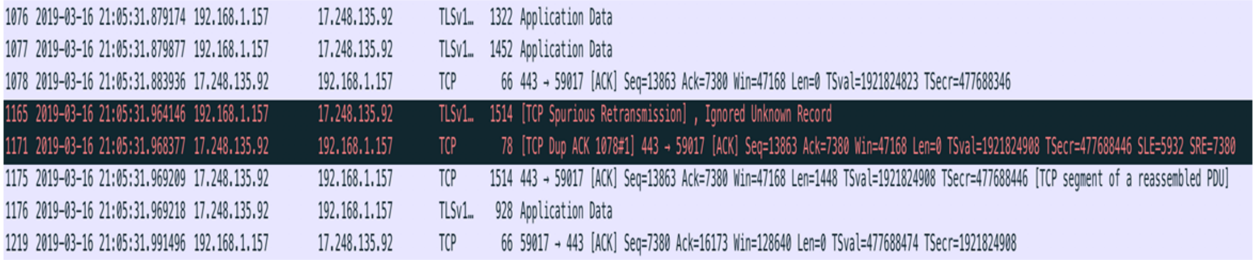
Status Code: ● 200 OK

Remote Address: 17.248.187.199:443

Referrer Policy: no-referrer-when-downgrade

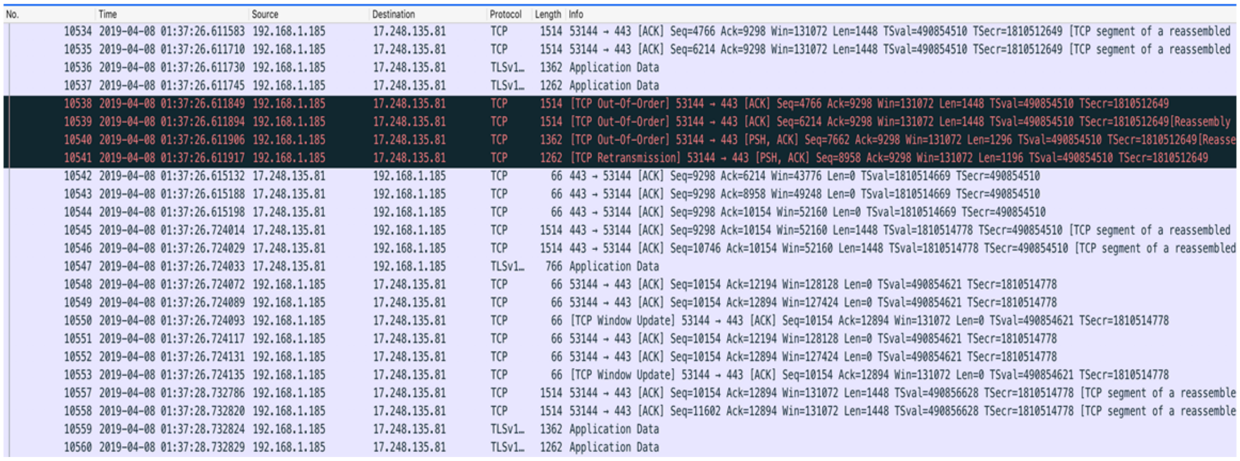
Log from Google Chrome network log showing refresh request sent by iCloud web app, captured April 8, 2019.

264. Claim 1 of the ’534 patent recites “send the fourth message to the first device over the first connection.” The ’534 accused products satisfy this limitation. For example, the Apple server forwards the second device’s location back to the first device.



```
1076 2019-03-16 21:05:31.879174 192.168.1.157  17.248.135.92  TLSv1… 1322 Application Data
1077 2019-03-16 21:05:31.879877 192.168.1.157  17.248.135.92  TLSv1… 1452 Application Data
1078 2019-03-16 21:05:31.883936 17.248.135.92  192.168.1.157  TCP      66 443 → 59017 [ACK] Seq=13863 Ack=7380 Win=47168 Len=0 TSval=1921824823 TSecr=477688346
1165 2019-03-16 21:05:31.964146 192.168.1.157  17.248.135.92  TLSv1… 1514 [TCP Spurious Retransmission] , Ignored Unknown Record
1171 2019-03-16 21:05:31.968377 17.248.135.92  192.168.1.157  TCP      78 [TCP Dup ACK 1078#1] 443 → 59017 [ACK] Seq=13863 Ack=7380 Win=47168 Len=0 TSval=1921824908 TSecr=477688446 SLE=5932 SRE=7380
1175 2019-03-16 21:05:31.969209 17.248.135.92  192.168.1.157  TCP    1514 443 → 59017 [ACK] Seq=13863 Ack=7380 Win=47168 Len=1448 TSval=1921824908 TSecr=477688446 [TCP segment of a reassembled PDU]
1176 2019-03-16 21:05:31.969218 17.248.135.92  192.168.1.157  TLSv1…  928 Application Data
1219 2019-03-16 21:05:31.991496 192.168.1.157  17.248.135.92  TCP      66 59017 → 443 [ACK] Seq=7380 Ack=16173 Win=128640 Len=0 TSval=477688474 TSecr=1921824908
```

Log showing that the first device, at 192.168.1.157 receives a response over the first connection which includes the second device’s location information, captured March 16, 2019.



```
No.         Time                       Source          Destination     Protocol Length Info
      10534 2019-04-08 01:37:26.611583 192.168.1.185   17.248.135.81   TCP      1514 53144 → 443 [ACK] Seq=4766 Ack=9298 Win=131072 Len=1448 TSval=490854510 TSecr=1810512649 [TCP segment of a reassembled
      10535 2019-04-08 01:37:26.611710 192.168.1.185   17.248.135.81   TCP      1514 53144 → 443 [ACK] Seq=6214 Ack=9298 Win=131072 Len=1448 TSval=490854510 TSecr=1810512649 [TCP segment of a reassembled
      10536 2019-04-08 01:37:26.611730 192.168.1.185   17.248.135.81   TLSv1…   1362 Application Data
      10537 2019-04-08 01:37:26.611745 192.168.1.185   17.248.135.81   TLSv1…   1262 Application Data
      10538 2019-04-08 01:37:26.611849 192.168.1.185   17.248.135.81   TCP      1514 [TCP Out-Of-Order] 53144 → 443 [ACK] Seq=4766 Ack=9298 Win=131072 Len=1448 TSval=490854510 TSecr=1810512649
      10539 2019-04-08 01:37:26.611894 192.168.1.185   17.248.135.81   TCP      1514 [TCP Out-Of-Order] 53144 → 443 [ACK] Seq=6214 Ack=9298 Win=131072 Len=1448 TSval=490854510 TSecr=1810512649[Reassembly
      10540 2019-04-08 01:37:26.611906 192.168.1.185   17.248.135.81   TCP      1362 [TCP Out-Of-Order] 53144 → 443 [PSH, ACK] Seq=7662 Ack=9298 Win=131072 Len=1296 TSval=490854510 TSecr=1810512649[Reasse
      10541 2019-04-08 01:37:26.611917 192.168.1.185   17.248.135.81   TCP      1262 [TCP Retransmission] 53144 → 443 [PSH, ACK] Seq=8958 Ack=9298 Win=131072 Len=1196 TSval=490854510 TSecr=1810512649
      10542 2019-04-08 01:37:26.615132 17.248.135.81   192.168.1.185   TCP        66 443 → 53144 [ACK] Seq=9298 Ack=6214 Win=43776 Len=0 TSval=1810514669 TSecr=490854510
      10543 2019-04-08 01:37:26.615188 17.248.135.81   192.168.1.185   TCP        66 443 → 53144 [ACK] Seq=9298 Ack=8958 Win=49248 Len=0 TSval=1810514669 TSecr=490854510
      10544 2019-04-08 01:37:26.615198 17.248.135.81   192.168.1.185   TCP        66 443 → 53144 [ACK] Seq=9298 Ack=10154 Win=52160 Len=0 TSval=1810514669 TSecr=490854510
      10545 2019-04-08 01:37:26.724014 17.248.135.81   192.168.1.185   TCP      1514 443 → 53144 [ACK] Seq=9298 Ack=10154 Win=52160 Len=1448 TSval=1810514778 TSecr=490854510 [TCP segment of a reassembled
      10546 2019-04-08 01:37:26.724029 17.248.135.81   192.168.1.185   TCP      1514 443 → 53144 [ACK] Seq=10746 Ack=10154 Win=52160 Len=1448 TSval=1810514778 TSecr=490854510 [TCP segment of a reassembled
      10547 2019-04-08 01:37:26.724033 17.248.135.81   192.168.1.185   TLSv1…    766 Application Data
      10548 2019-04-08 01:37:26.724072 192.168.1.185   17.248.135.81   TCP        66 53144 → 443 [ACK] Seq=10154 Ack=12194 Win=128128 Len=0 TSval=490854621 TSecr=1810514778
      10549 2019-04-08 01:37:26.724089 192.168.1.185   17.248.135.81   TCP        66 53144 → 443 [ACK] Seq=10154 Ack=12894 Win=127424 Len=0 TSval=490854621 TSecr=1810514778
      10550 2019-04-08 01:37:26.724093 192.168.1.185   17.248.135.81   TCP        66 [TCP Window Update] 53144 → 443 [ACK] Seq=10154 Ack=12894 Win=131072 Len=0 TSval=490854621 TSecr=1810514778
      10551 2019-04-08 01:37:26.724117 192.168.1.185   17.248.135.81   TCP        66 53144 → 443 [ACK] Seq=10154 Ack=12194 Win=128128 Len=0 TSval=490854621 TSecr=1810514778
      10552 2019-04-08 01:37:26.724131 192.168.1.185   17.248.135.81   TCP        66 53144 → 443 [ACK] Seq=10154 Ack=12894 Win=127424 Len=0 TSval=490854621 TSecr=1810514778
      10553 2019-04-08 01:37:26.724135 192.168.1.185   17.248.135.81   TCP        66 [TCP Window Update] 53144 → 443 [ACK] Seq=10154 Ack=12894 Win=131072 Len=0 TSval=490854621 TSecr=1810514778
      10557 2019-04-08 01:37:28.732786 192.168.1.185   17.248.135.81   TCP      1514 53144 → 443 [ACK] Seq=10154 Ack=12894 Win=131072 Len=1448 TSval=490856628 TSecr=1810514778 [TCP segment of a reassemble
      10558 2019-04-08 01:37:28.732820 192.168.1.185   17.248.135.81   TCP      1514 53144 → 443 [ACK] Seq=11602 Ack=12894 Win=131072 Len=1448 TSval=490856628 TSecr=1810514778 [TCP segment of a reassemble
      10559 2019-04-08 01:37:28.732824 192.168.1.185   17.248.135.81   TLSv1…   1362 Application Data
      10560 2019-04-08 01:37:28.732829 192.168.1.185   17.248.135.81   TLSv1…   1262 Application Data
```

Log showing that the web browser, at 192.168.1.185 receives a response over the first connection which includes the second device's location information, captured April 8, 2019.

265. Thus, as illustrated above, the '534 accused products directly infringe one or more claims of the '534 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the '534 patent.

266. Apple indirectly infringes the '534 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple's customers and end-users, in this district and elsewhere in the United States. For example, Apple's customers and end-users directly infringe through their use of the inventions claimed in the '534 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the '534 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the '534 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of

infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products' capability in an infringing manner, including using the Find My Friends and Find My iPhone applications. *See, e.g.*, https://support.apple.com/kb/PH2698?viewlocale=en_US&locale=en_US; https://support.apple.com/en-us/HT201493. As a result of Apple's inducement, Apple's customers and end-users use the Apple products in the way Apple intends and directly infringe the '534 patent. Apple performs these affirmative acts with knowledge of the '534 patent and with the intent, or willful blindness, that the induced acts directly infringe the '534 patent. Apple has had knowledge and notice of the '534 patent at least as of the filing of this complaint.

267. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

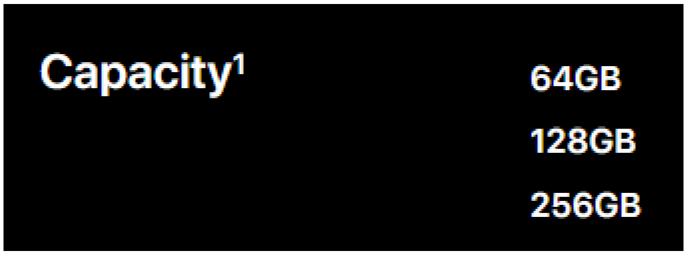
COUNT XV: PATENT INFRINGEMENT OF THE '771 PATENT

268. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

269. Apple infringes (literally and/or under the Doctrine of Equivalents) the '771 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the '771 patent including at least the '771 accused products noted above. In addition, Apple derives revenue from the activities relating to the '771 accused products.

270. For example and as shown below, the '771 accused products infringe at least claim 1 of the '771 patent. For example, to the extent the preamble is limiting, the '771 accused products satisfy "a non-transitory computer-readable medium storing instructions to be implemented by a first computer having a processor, wherein the instructions, when executed by

the processor, cause the first computer to perform steps comprising:" As shown below by the exemplary evidence, the '771 accused products include a non-transitory computer-readable medium storing instruction to be implemented by a first computer having a processor, wherein the instructions, when executed by the processor, cause the first computer to perform steps.



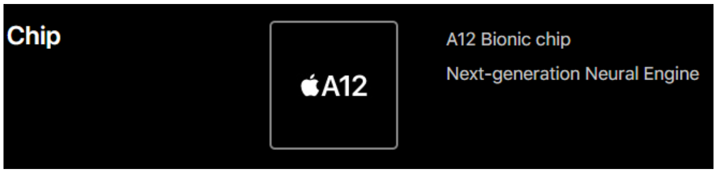
Capacity[1]

64GB

128GB

256GB

https://www.apple.com/iphone-xr/specs/



Chip

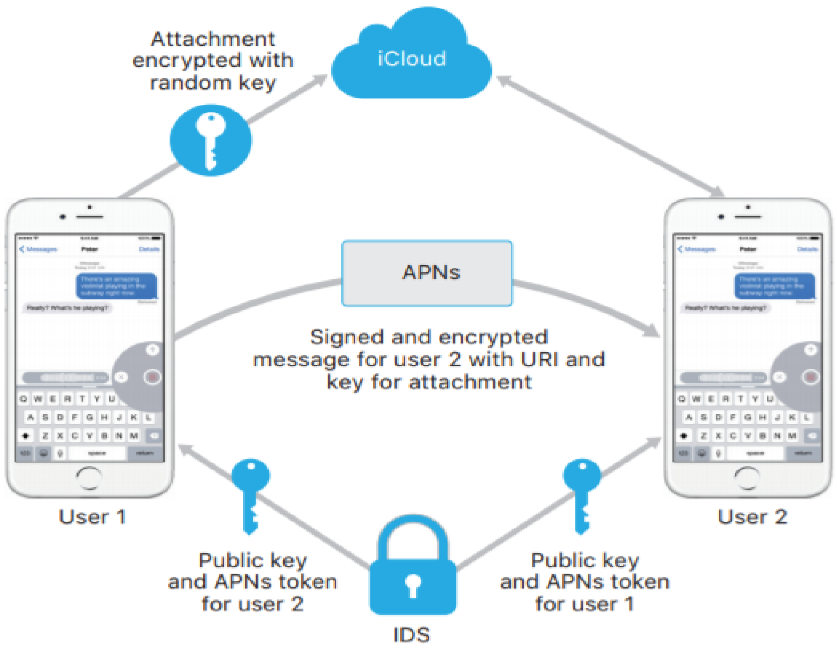
A12

A12 Bionic chip

Next-generation Neural Engine

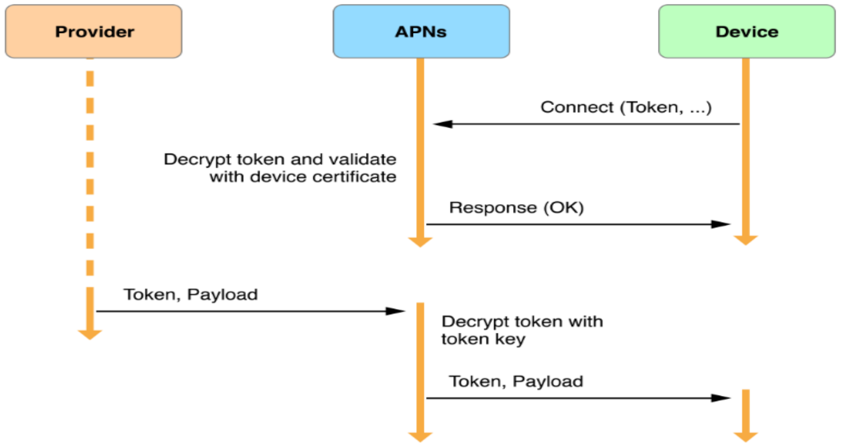
https://www.apple.com/iphone-xr/specs/

271. Apple's iPhones, iPads, and macs are capable of communicating with an intermediary server, such as server having APNs functionality, to send messages, including iMessages, to another device. Apple's server supporting iCloud functionality, such as for example Find My Friends and Find My iPhone functionality, is also capable of communicating with an intermediary server, such as a server having APNs functionality, to send messages to another device using an APNs-issued token

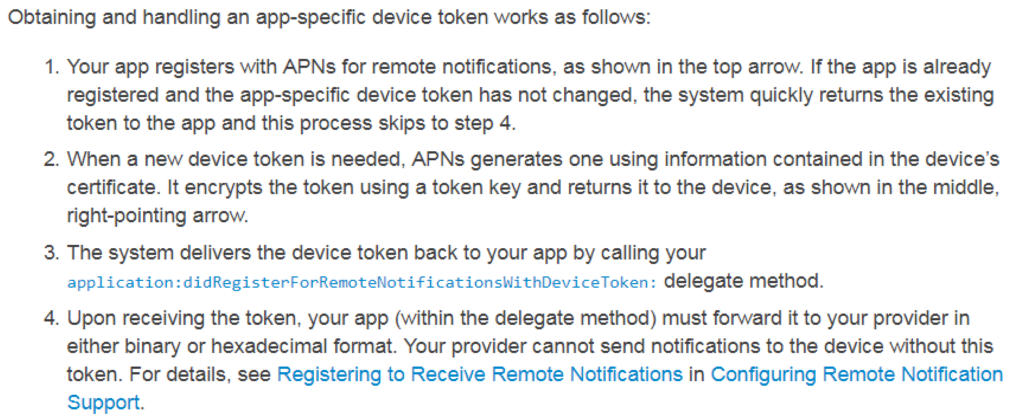

Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

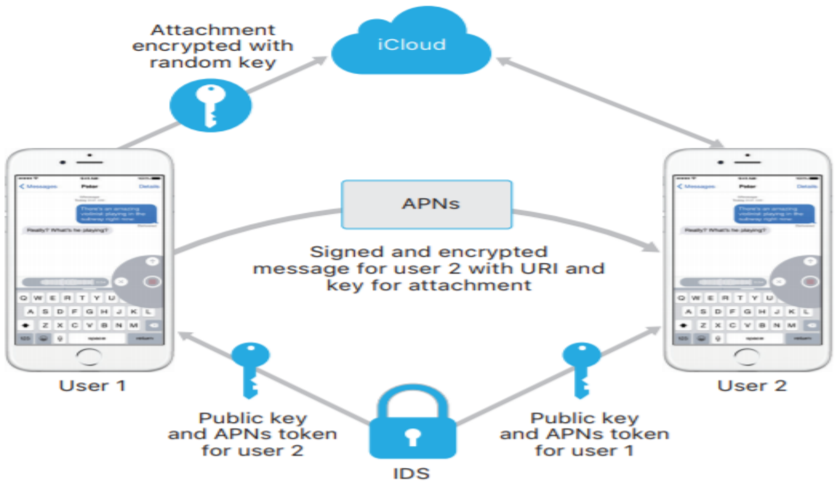


Obtaining and handling an app-specific device token works as follows:

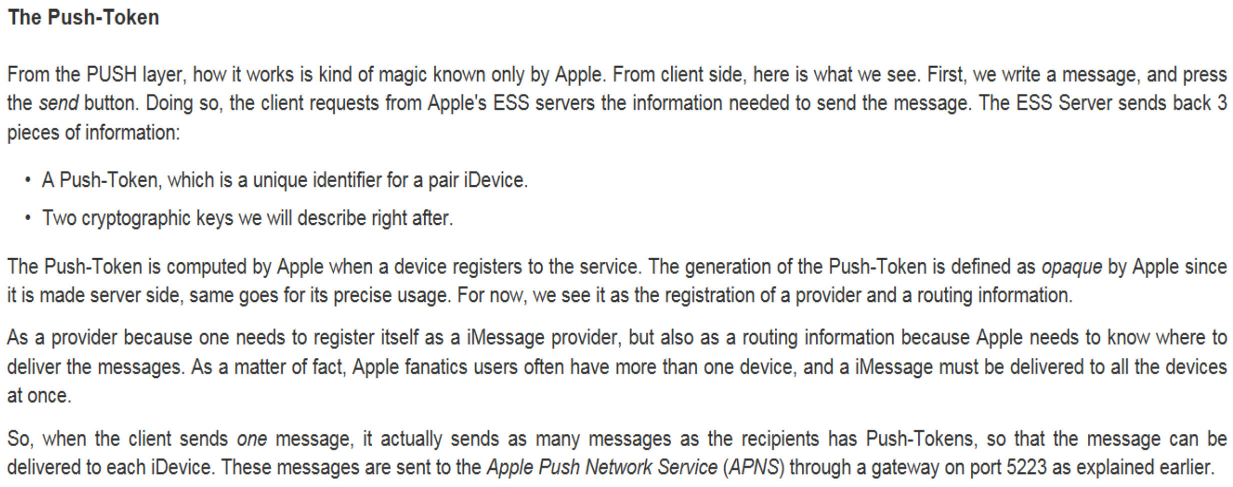
1. Your app registers with APNs for remote notifications, as shown in the top arrow. If the app is already registered and the app-specific device token has not changed, the system quickly returns the existing token to the app and this process skips to step 4.
2. When a new device token is needed, APNs generates one using information contained in the device's certificate. It encrypts the token using a token key and returns it to the device, as shown in the middle, right-pointing arrow.
3. The system delivers the device token back to your app by calling your `application:didRegisterForRemoteNotificationsWithDeviceToken:` delegate method.
4. Upon receiving the token, your app (within the delegate method) must forward it to your provider in either binary or hexadecimal format. Your provider cannot send notifications to the device without this token. For details, see Registering to Receive Remote Notifications in Configuring Remote Notification Support.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

272. Claim 1 of the '771 patent recites "receiving a token issued by an intermediary server." The '771 accused products receive a token issued by an intermediary server.


Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf



The Push-Token

From the PUSH layer, how it works is kind of magic known only by Apple. From client side, here is what we see. First, we write a message, and press the *send* button. Doing so, the client requests from Apple's ESS servers the information needed to send the message. The ESS Server sends back 3 pieces of information:

- A Push-Token, which is a unique identifier for a pair iDevice.
- Two cryptographic keys we will describe right after.

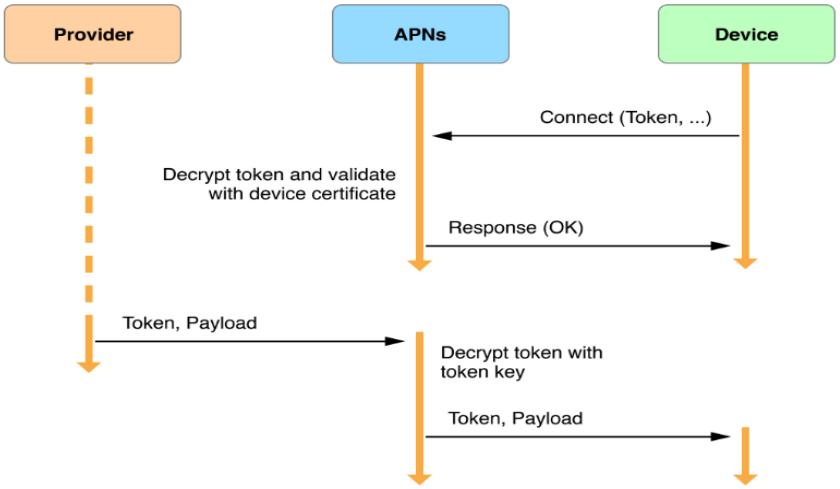
The Push-Token is computed by Apple when a device registers to the service. The generation of the Push-Token is defined as *opaque* by Apple since it is made server side, same goes for its precise usage. For now, we see it as the registration of a provider and a routing information.

As a provider because one needs to register itself as a iMessage provider, but also as a routing information because Apple needs to know where to deliver the messages. As a matter of fact, Apple fanatics users often have more than one device, and a iMessage must be delivered to all the devices at once.

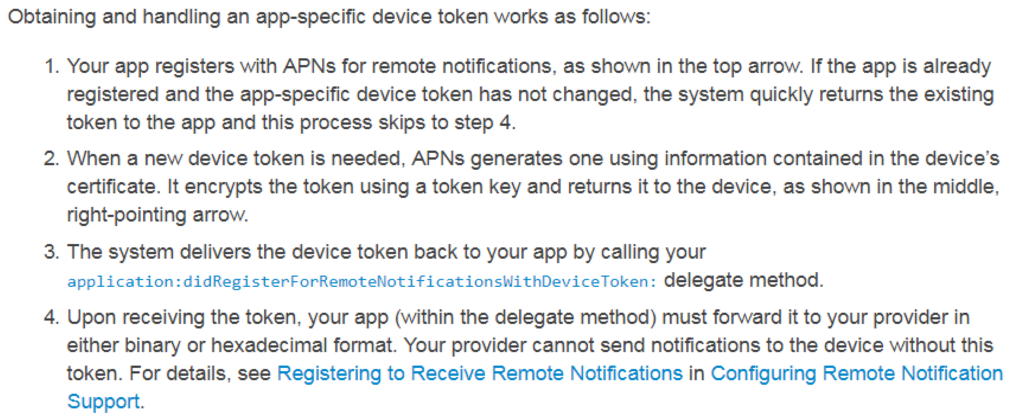
So, when the client sends *one* message, it actually sends as many messages as the recipients has Push-Tokens, so that the message can be delivered to each iDevice. These messages are sent to the *Apple Push Network Service* (*APNS*) through a gateway on port 5223 as explained earlier.

https://blog.quarkslab.com/imessage-privacy.html

Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1



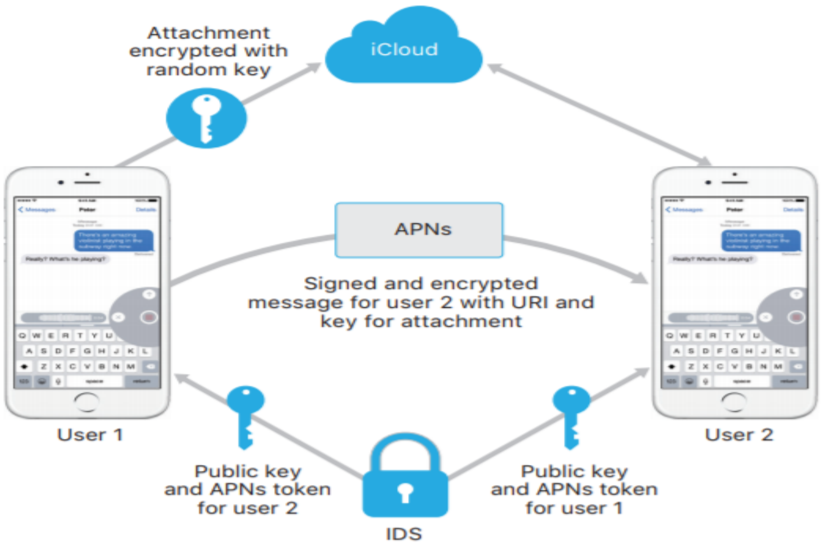
Obtaining and handling an app-specific device token works as follows:

1. Your app registers with APNs for remote notifications, as shown in the top arrow. If the app is already registered and the app-specific device token has not changed, the system quickly returns the existing token to the app and this process skips to step 4.
2. When a new device token is needed, APNs generates one using information contained in the device's certificate. It encrypts the token using a token key and returns it to the device, as shown in the middle, right-pointing arrow.
3. The system delivers the device token back to your app by calling your `application:didRegisterForRemoteNotificationsWithDeviceToken:` delegate method.
4. Upon receiving the token, your app (within the delegate method) must forward it to your provider in either binary or hexadecimal format. Your provider cannot send notifications to the device without this token. For details, see Registering to Receive Remote Notifications in Configuring Remote Notification Support.

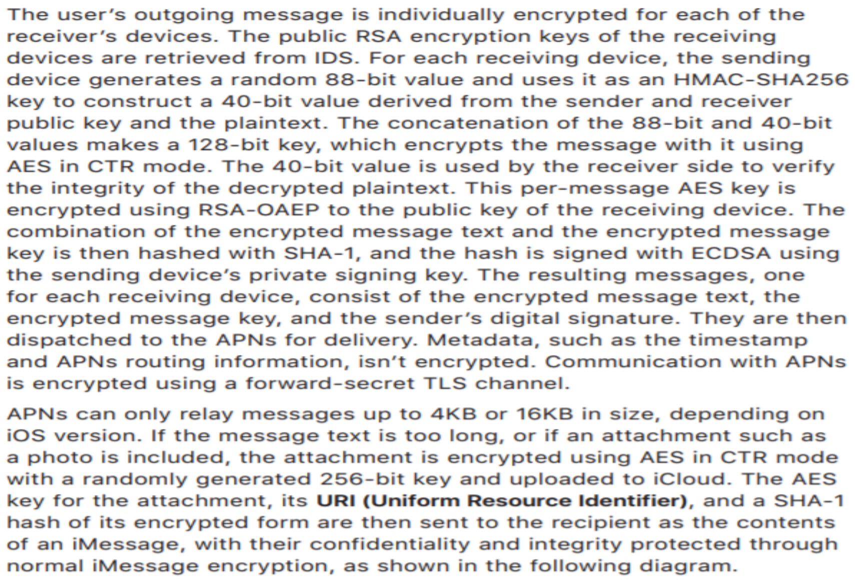
https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

273. Claim 1 of the '771 patent recites "transmitting a transaction message comprising payload data to the intermediary server, wherein the payload data is transmitted to a second computer by the intermediary server based on the token and the intermediary server is coupled to the second computer over a mobile network." The '771 accused products transmit a transaction message comprising payload data to the intermediary server, wherein the payload data is

transmitted to a second computer by the intermediary server based on the token and the intermediary server is coupled to the second computer over a mobile network.


Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

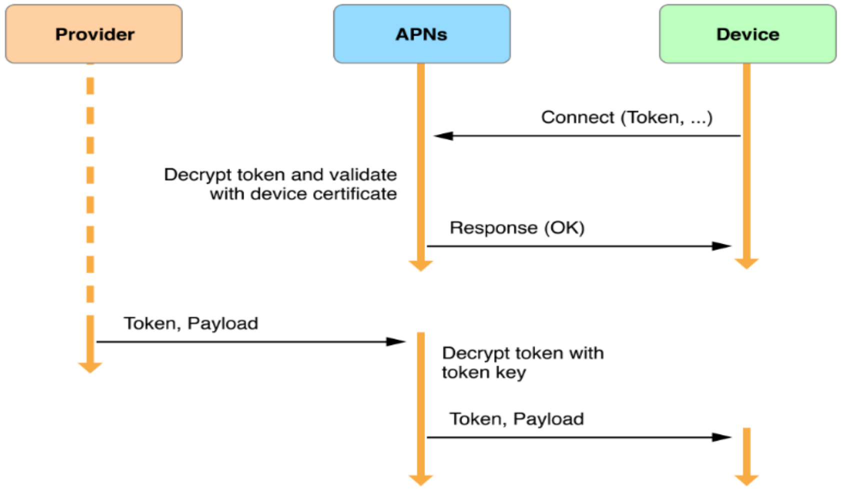


The user's outgoing message is individually encrypted for each of the receiver's devices. The public RSA encryption keys of the receiving devices are retrieved from IDS. For each receiving device, the sending device generates a random 88-bit value and uses it as an HMAC-SHA256 key to construct a 40-bit value derived from the sender and receiver public key and the plaintext. The concatenation of the 88-bit and 40-bit values makes a 128-bit key, which encrypts the message with it using AES in CTR mode. The 40-bit value is used by the receiver side to verify the integrity of the decrypted plaintext. This per-message AES key is encrypted using RSA-OAEP to the public key of the receiving device. The combination of the encrypted message text and the encrypted message key is then hashed with SHA-1, and the hash is signed with ECDSA using the sending device's private signing key. The resulting messages, one for each receiving device, consist of the encrypted message text, the encrypted message key, and the sender's digital signature. They are then dispatched to the APNs for delivery. Metadata, such as the timestamp and APNs routing information, isn't encrypted. Communication with APNs is encrypted using a forward-secret TLS channel.

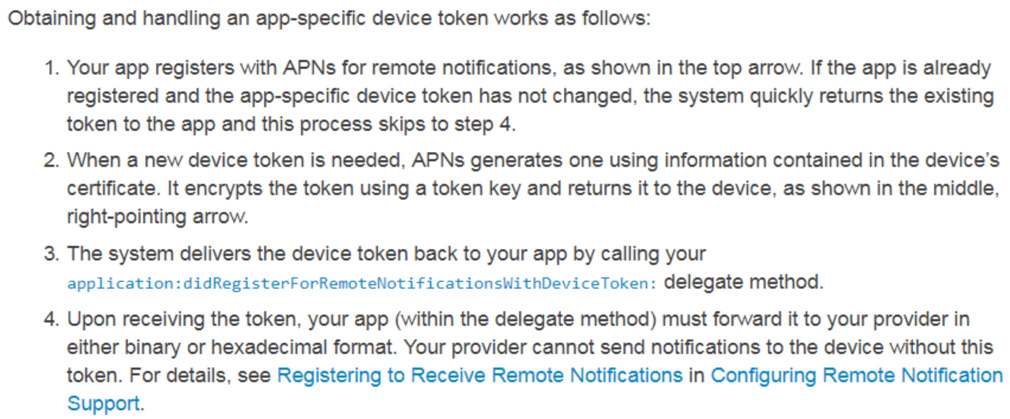
APNs can only relay messages up to 4KB or 16KB in size, depending on iOS version. If the message text is too long, or if an attachment such as a photo is included, the attachment is encrypted using AES in CTR mode with a randomly generated 256-bit key and uploaded to iCloud. The AES key for the attachment, its **URI (Uniform Resource Identifier)**, and a SHA-1 hash of its encrypted form are then sent to the recipient as the contents of an iMessage, with their confidentiality and integrity protected through normal iMessage encryption, as shown in the following diagram.

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

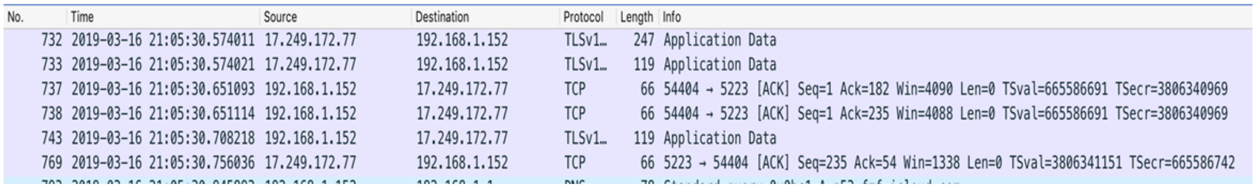

Obtaining and handling an app-specific device token works as follows:

1. Your app registers with APNs for remote notifications, as shown in the top arrow. If the app is already registered and the app-specific device token has not changed, the system quickly returns the existing token to the app and this process skips to step 4.
2. When a new device token is needed, APNs generates one using information contained in the device's certificate. It encrypts the token using a token key and returns it to the device, as shown in the middle, right-pointing arrow.
3. The system delivers the device token back to your app by calling your `application:didRegisterForRemoteNotificationsWithDeviceToken:` delegate method.
4. Upon receiving the token, your app (within the delegate method) must forward it to your provider in either binary or hexadecimal format. Your provider cannot send notifications to the device without this token. For details, see Registering to Receive Remote Notifications in Configuring Remote Notification Support.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

```
default  21:05:30.662300 -0400      apsd     2019-03-16   21:05:30   -0400   apsd[85]:   <APSCourier:   0x11bd0d530>: Outstanding data received: <0a7d8101 42147f29 2acab37b ed771b36 b09e612e 395eee13 a69a0480 05a80106 a7158c99 7c44677e 47078003 3f127b22 73657276 6572436f 6e746578 74223a7b 22746170 53656e64 5453223a 22323031 392d3033 2d313754 30313a30 353a3330 2e353331 5a222c22 74617053 656e6443 6f6e7465 7874223a 22666d66 227d7d0d 020181> (length 131) onInterface: NonCellular. Connected on 1 interfaces.

default  21:05:30.665491 -0400      apsd     2019-03-16 21:05:30 -0400 apsd[85]: <APSCourier: 0x11bd0d530>: Stream processing: complete yes, invalid no, length parsed 127, parameters {

APSProtocolCommand = 10;

APSProtocolMessageExpiry = "1970-01-01 18:12:15 +0000";

APSProtocolMessageID = <00000000>;

APSProtocolMessageTimestamp = 1552784730538606151;

APSProtocolPayload = <7b227365 72766572 436f6e74 65787422 3a7b2274 61705365 6e645453 223a2232 3031392d 30332d31 37543031 3a30353a 33302e35 33315a22 2c227461 7053656e 64436f6e 74657874 223a2266 6d66227d 7d>;

APSProtocolToken = <6dc220e2 a40b027a d01c77ce 7df3f958 fdbb9ec5 bf218587 fc444b14 e2e9c4b1>;
APSProtocolTopicHash = <7f292aca b37bed77 1b36b09e 612e395e ee13a69a>;

}

default  21:05:30.667717 -0400      apsd     2019-03-16  21:05:30  -0400  apsd[85]:  copyTokenForDomain  push.apple.com
(null)

default  21:05:30.681194 -0400      apsd     2019-03-16  21:05:30  -0400  apsd[85]:  copyTokenForDomain  push.apple.com
(null)

default  21:05:30.693112 -0400      apsd     2019-03-16  21:05:30  -0400  apsd[85]:  copyTokenForDomain  push.apple.com
(null)

default  21:05:30.700560 -0400      apsd     2019-03-16  21:05:30  -0400  apsd[85]:  copyTokenForDomain  push.apple.com
(null)

default  21:05:30.706677 -0400      apsd     2019-03-16   21:05:30   -0400   apsd[85]:   <APSPushHistory:   0x11bd405d0>
timestampForTopic?  com.apple.mobileme.fmf  token  <6dc220e2  a40b027a  d01c77ce  7df3f958  fdbb9ec5  bf218587  fc444b14
e2e9c4b1>

default  21:05:30.707061 -0400      apsd     2019-03-16  21:05:30  -0400  apsd[85]:  copyTokenForDomain  push.apple.com
,7f292affffffcaffffffb37bffffffed771b36ffffffb0ffffff9e612e395effffffee13ffffffa6ffffff9a

default  21:05:30.713395 -0400      apsd     2019-03-16   21:05:30   -0400   apsd[85]:   <APSPushHistory:   0x11bd405d0>
timestampForTopic? com.apple.mobileme.fmf token (null)

default  21:05:30.714018 -0400      apsd     2019-03-16   21:05:30   -0400   apsd[85]:   <APSPushHistory:   0x11bd405d0>
receivedPushWithTopic  com.apple.mobileme.fmf  token <6dc220e2  a40b027a  d01c77ce  7df3f958  fdbb9ec5  bf218587  fc444b14
e2e9c4b1> payload <7b227365 72766572 436f6e74 65787422 3a7b2274 61705365 6e645453 223a2232 3031392d 30332d31 37543031
3a30353a  33302e35  33315a22  2c227461  7053656e  64436f6e  74657874  223a2266  6d66227d  7d>  timestamp  2019-03-17  01:05:30
+0000
```

iOS log from second device showing that apsd – the APNs client on the second device – receives a push notification with topic "com.apple.mobileme.fmf," captured March 16, 2019.



No.	Time	Source	Destination	Protocol	Length	Info
732	2019-03-16 21:05:30.574011	17.249.172.77	192.168.1.152	TLSv1…	247	Application Data
733	2019-03-16 21:05:30.574021	17.249.172.77	192.168.1.152	TLSv1…	119	Application Data
737	2019-03-16 21:05:30.651093	192.168.1.152	17.249.172.77	TCP	66	54404 → 5223 [ACK] Seq=1 Ack=182 Win=4090 Len=0 TSval=665586691 TSecr=3806340969
738	2019-03-16 21:05:30.651114	192.168.1.152	17.249.172.77	TCP	66	54404 → 5223 [ACK] Seq=1 Ack=235 Win=4088 Len=0 TSval=665586691 TSecr=3806340969
743	2019-03-16 21:05:30.708218	192.168.1.152	17.249.172.77	TLSv1…	119	Application Data
769	2019-03-16 21:05:30.756036	17.249.172.77	192.168.1.152	TCP	66	5223 → 54404 [ACK] Seq=235 Ack=54 Win=1338 Len=0 TSval=3806341151 TSecr=665586742

Packet log showing transmission of push notification from APNs at 17.249.172.77

274. Thus, as illustrated above, the '771 accused products directly infringe one or more claims of the '771 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the '771 patent.

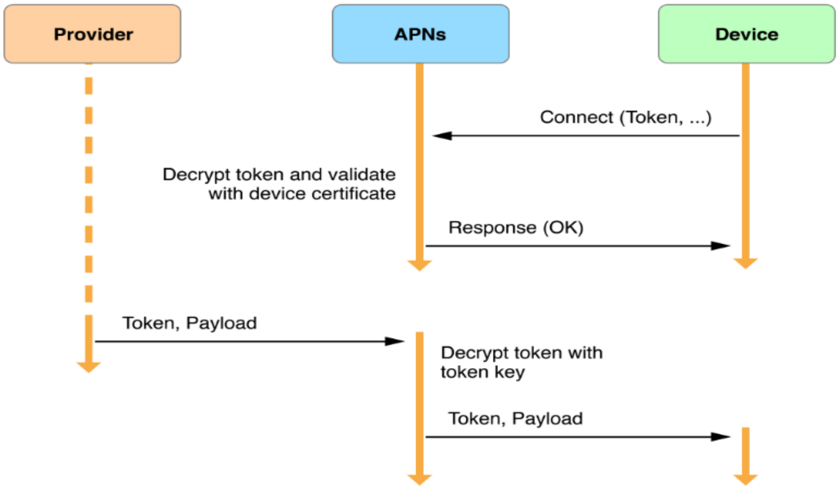
275. Apple indirectly infringes the '771 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple's customers and end-users, in this district and elsewhere in the United States. For example, Apple's customers and end-users directly infringe through their use of the inventions claimed in the '771 patent. Apple

induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’771 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the ’771 accused products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products’ capability in an infringing manner, including including through the use of Apple’s APNs functionality as well as iMessage. *See, e.g.,* https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1; https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf.. As a result of Apple’s inducement, Apple’s customers and end-users use the Apple products in the way Apple intends and directly infringe the ’771 patent. Apple performs these affirmative acts with knowledge of the ’771 patent and with the intent, or willful blindness, that the induced acts directly infringe the ’771 patent. Apple has had knowledge and notice of the ’771 patent at least as of the filing of this complaint.

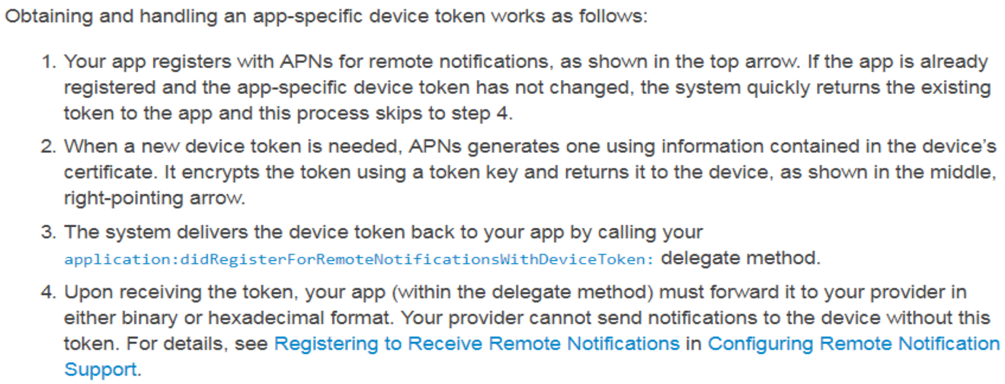
276. By way of one example of Apple’s indirect infringement, Apple indirectly infringes because it encourages infringing use by third party first computers, such as computers associated with app providers, which are configured for receiving a token issued by an

intermediary server, such as an app-specific token issued by Apple's server supporting APNs functionality, and transmitting a transaction message comprising payload data to the Apple server supporting APNs functionality, wherein the payload data is transmitted to a second computer, such as an Apple iOS device by the intermediary server based on the app-specific token and the intermediary server is coupled to the second computer over a mobile network.

Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1


Obtaining and handling an app-specific device token works as follows:

1. Your app registers with APNs for remote notifications, as shown in the top arrow. If the app is already registered and the app-specific device token has not changed, the system quickly returns the existing token to the app and this process skips to step 4.
2. When a new device token is needed, APNs generates one using information contained in the device's certificate. It encrypts the token using a token key and returns it to the device, as shown in the middle, right-pointing arrow.
3. The system delivers the device token back to your app by calling your `application:didRegisterForRemoteNotificationsWithDeviceToken:` delegate method.
4. Upon receiving the token, your app (within the delegate method) must forward it to your provider in either binary or hexadecimal format. Your provider cannot send notifications to the device without this token. For details, see Registering to Receive Remote Notifications in Configuring Remote Notification Support.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

277. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

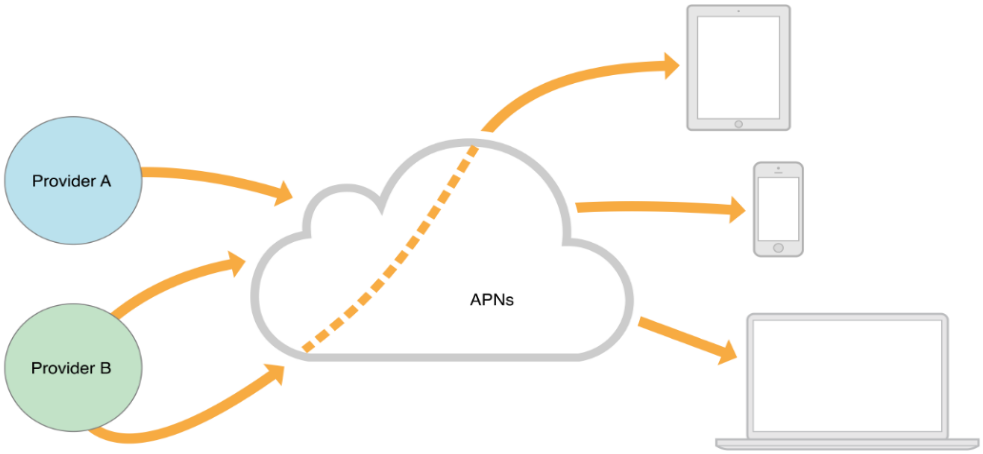
COUNT XVI: PATENT INFRINGEMENT OF THE ’962 PATENT

278. Plaintiff incorporates by reference the preceding paragraphs as though fully set forth herein.

279. Apple infringes (literally and/or under the Doctrine of Equivalents) the ’962 patent by making, using, offering for sale, selling, and/or importing into the United States products and/or methods covered by one or more claims of the ’962 patent including at least the ’962 accused products noted above. In addition, Apple derives revenue from the activities relating to the ’962 accused products.

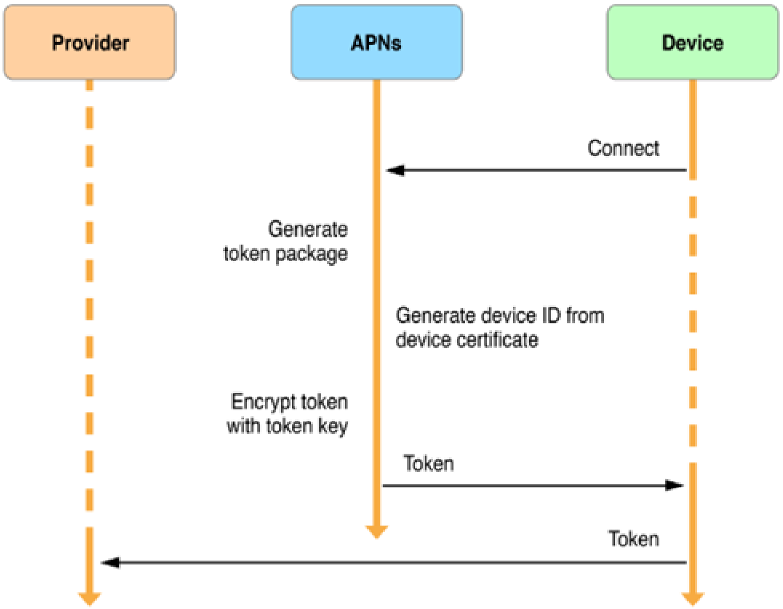
280. For example and as shown below, the ’962 accused products infringe at least claim 1 of the ’962 patent. For example, to the extent the preamble is limiting, the ’962 accused products satisfy a “server, comprising: a memory and a processor, the server communicatively coupled to a network and a plurality of data stores, wherein the server is configured to:” For example as described in the exemplary documentation below, Apple’s server supporting APNs functionality provides access to data stores, for example, related to app providers.

Figure 6-2 Pushing remote notifications from multiple providers to multiple devices



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html

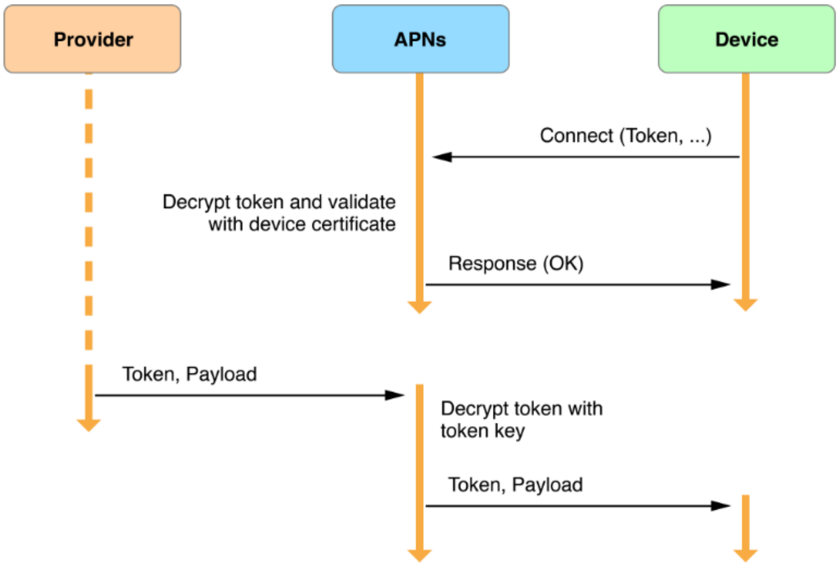
Figure 6-6 Managing the device token



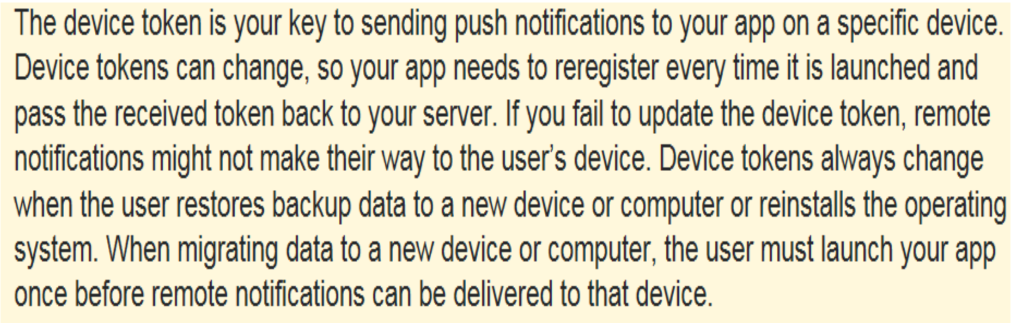
https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

281. Claim 1 of the ’962 patent recites “send a first identifier to a client device upon the client device communicating with the server over a connection.” The ’962 accused products satisfy this limitation. For example, the exemplary documentation below shows Apple’s server sends a first identifier for a client device upon the client device communicating with the server over a connection.

Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1


The device token is your key to sending push notifications to your app on a specific device. Device tokens can change, so your app needs to reregister every time it is launched and pass the received token back to your server. If you fail to update the device token, remote notifications might not make their way to the user's device. Device tokens always change when the user restores backup data to a new device or computer or reinstalls the operating system. When migrating data to a new device or computer, the user must launch your app once before remote notifications can be delivered to that device.

https://stackoverflow.com/questions/6652242/does-the-apns-device-token-ever-change-once-created

```
default 12:32:08.827534 -0400 apsd 2019-03-25 12:32:08 -0400 apsd[85]: <APSCourier: 0x11bd0d530>: Connected to courier 36-courier.push.apple.com (17.249.76.5) connection: <APSCourierConnection: 0x11bd21f70> usingPackedFormat YES secureHandshakeEnabled YES onInterface: NonCellular
default 12:32:08.827619 -0400 apsd 2019-03-25 12:32:08 -0400 apsd[85]: copyTokenForDomain push.apple.com (null)
default 12:32:08.832767 -0400 apsd 2019-03-25 12:32:08 -0400 apsd[85]: copyTokenForDomain push.apple.com (null)
default 12:32:08.838639 -0400 apsd 2019-03-25 12:32:08 -0400 apsd[85]: copyTokenForDomain push.apple.com (null)
default 12:32:08.839255 -0400 apsd 2019-03-25 12:32:08 -0400 apsd[85]: <APSCourier: 0x11bd0d530> Sending presence message for deviceToken with token YES on interface: NonCellular tokenName: systemToken
```

iPhone log, captured March 25, 2019

```
default    12:32:09.008560 -0400    apsd    2019-03-25 12:32:08 -0400 apsd[85]:
<APSCourier: 0x11bd0d530>: Sending filter message for: token type systemToken, token:
<6dc220e2 a40b027a d01c77ce 7df3f958 fdbb9ec5 bf218587 fc444b14 e2e9c4b1>, enabled
hashes {
<2038739c 2e580f36 13e5e2ed cdb6511f 610988d0> = "com.apple.icloud-
container.com.apple.willowd";
<61b1ed7e 04116ffc a2b48ccf fa9111a9 f4cb772a> = "com.tencent.xin.voip";
<e2225610 d1b716ac 87094ad0 2abd95cb 727975fb> = "com.apple.icloud-
container.com.apple.cloudpaird";
<f690f1b3 413e1123 a4e624b5 36e6ddba 55d6b567> = "com.google.GVDialer.voip";
<3218784d 5fe3f32e 93cfaa9d 8cf6f6cf 39e6d522> = "com.vzw.pushnotification";
<ed7660ad 372f623f be99baa4 9ea5b97f d98195b2> = "com.spothero.spothero.voip";
<94a9e606 83653fdd e5192867 3963d5a0 fca0e8c9> = "ph.telegra.Telegraph.voip";
<efd8f204 92ab0112 9eef9fb0 a834cecf 23d8143c> = "com.apple.icloud-
container.com.apple.accessibility.heard";
<7496b146 4c8a18ec b9c487fa 8923aa36 8a0bde4f> = "com.mastercard.tokenizati<…>
```

iPhone log, captured March 25, 2019

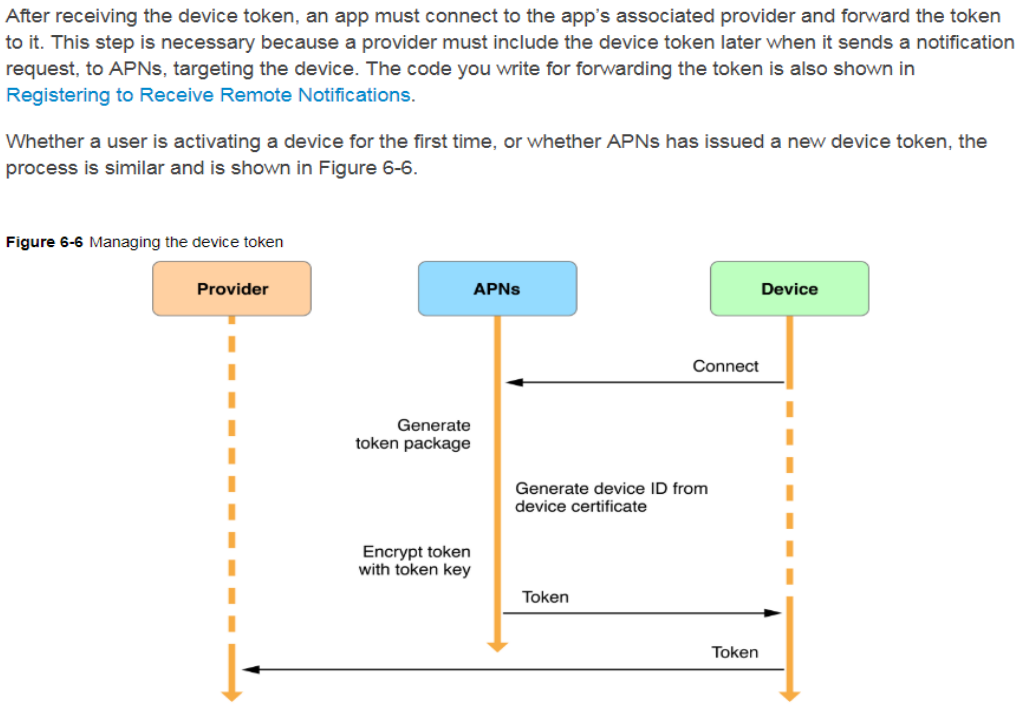
```
default    16:23:20.453553 -0400    apsd    2019-03-25 16:23:19 -0400 apsd[87]:
<APSCourier: 0x10021a020>: Stream processing: complete yes, invalid no, length parsed 60,
parameters {
APSProtocolAppTokenGenerateResponse = 0;
APSProtocolAppTokenGenerateResponseToken = <f8474454 9fcb9c9f ed0185d9 375577f1
a84291b4 c2f47759 6753f645 5f6744fa>;
APSProtocolAppTokenGenerateResponseTopicHash = <3218784d 5fe3f32e 93cfaa9d 8cf6f6cf
39e6d522>;
   APSProtocolCommand = 18;
```

iPhone log, captured March 25, 2019

```
default    12:32:09.008560 -0400    apsd    2019-03-25 12:32:08 -0400 apsd[85]:
<APSCourier: 0x11bd0d530>: Sending filter message for: token type systemToken, token:
<6dc220e2 a40b027a d01c77ce 7df3f958 fdbb9ec5 bf218587 fc444b14 e2e9c4b1>, enabled
hashes {
<2038739c 2e580f36 13e5e2ed cdb6511f 610988d0> = "com.apple.icloud-
container.com.apple.willowd";
<61b1ed7e 04116ffc a2b48ccf fa9111a9 f4cb772a> = "com.tencent.xin.voip"; <e2225610
d1b716ac 87094ad0 2abd95cb 727975fb> = "com.apple.icloud-container.com.apple.cloudpaird";
<f690f1b3 413e1123 a4e624b5 36e6ddba 55d6b567> = "com.google.GVDialer.voip";
<3218784d 5fe3f32e 93cfaa9d 8cf6f6cf 39e6d522> = "com.vzw.pushnotification";
```

iPhone log, captured March 25, 2019

282. Claim 1 of the ’962 patent recites “receive registration information from the client device.” The ’962 accused products satisfy this limitation. The ’962 accused products receive registration information from the client.



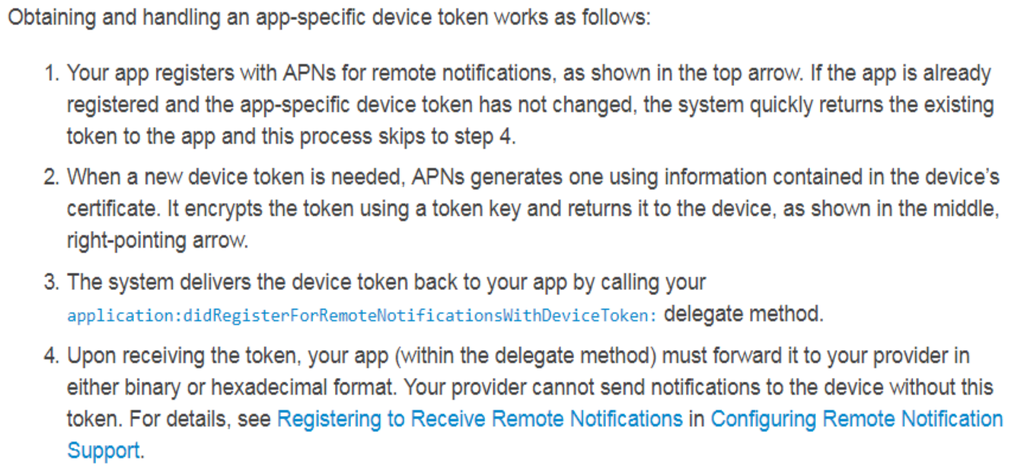
After receiving the device token, an app must connect to the app’s associated provider and forward the token to it. This step is necessary because a provider must include the device token later when it sends a notification request, to APNs, targeting the device. The code you write for forwarding the token is also shown in Registering to Receive Remote Notifications.

Whether a user is activating a device for the first time, or whether APNs has issued a new device token, the process is similar and is shown in Figure 6-6.

Figure 6-6 Managing the device token

Provider
APNs
Device
Connect
Generate token package
Generate device ID from device certificate
Encrypt token with token key
Token
Token

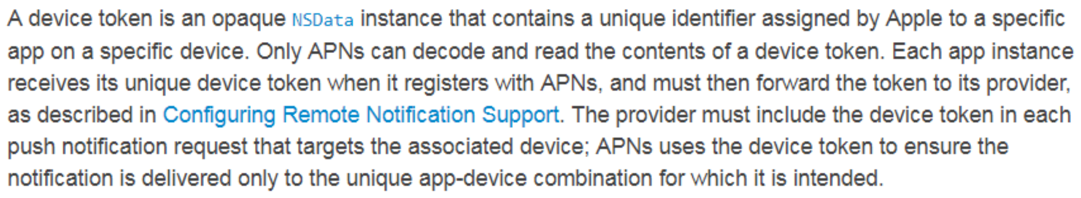
https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1



Obtaining and handling an app-specific device token works as follows:

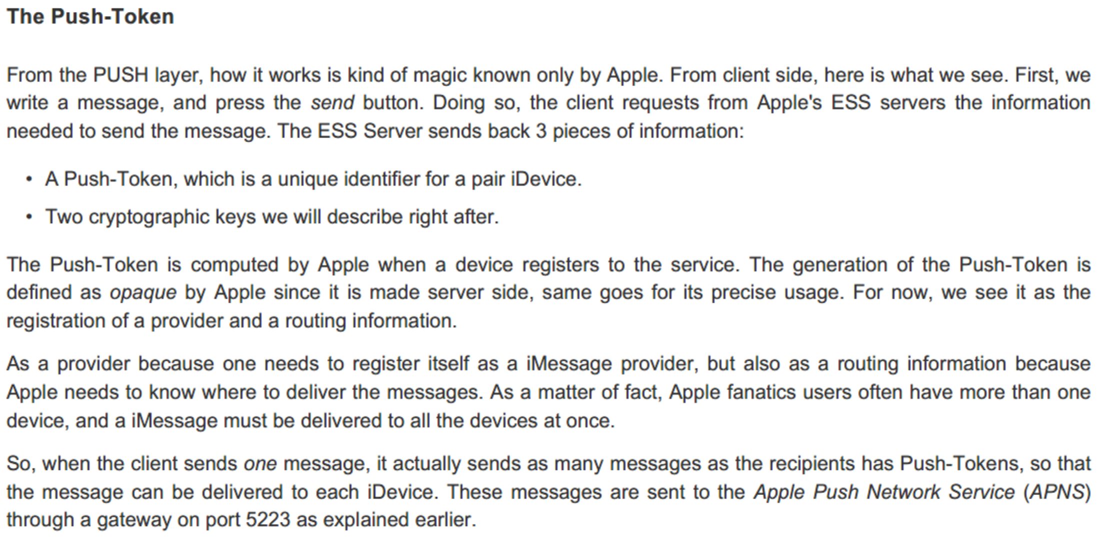
1. Your app registers with APNs for remote notifications, as shown in the top arrow. If the app is already registered and the app-specific device token has not changed, the system quickly returns the existing token to the app and this process skips to step 4.
2. When a new device token is needed, APNs generates one using information contained in the device’s certificate. It encrypts the token using a token key and returns it to the device, as shown in the middle, right-pointing arrow.
3. The system delivers the device token back to your app by calling your `application:didRegisterForRemoteNotificationsWithDeviceToken:` delegate method.
4. Upon receiving the token, your app (within the delegate method) must forward it to your provider in either binary or hexadecimal format. Your provider cannot send notifications to the device without this token. For details, see Registering to Receive Remote Notifications in Configuring Remote Notification Support.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1



A device token is an opaque NSData instance that contains a unique identifier assigned by Apple to a specific app on a specific device. Only APNs can decode and read the contents of a device token. Each app instance receives its unique device token when it registers with APNs, and must then forward the token to its provider, as described in Configuring Remote Notification Support. The provider must include the device token in each push notification request that targets the associated device; APNs uses the device token to ensure the notification is delivered only to the unique app-device combination for which it is intended.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1



The Push-Token

From the PUSH layer, how it works is kind of magic known only by Apple. From client side, here is what we see. First, we write a message, and press the *send* button. Doing so, the client requests from Apple's ESS servers the information needed to send the message. The ESS Server sends back 3 pieces of information:

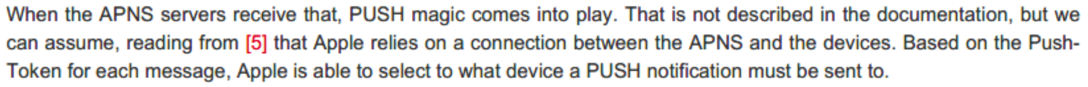
- A Push-Token, which is a unique identifier for a pair iDevice.
- Two cryptographic keys we will describe right after.

The Push-Token is computed by Apple when a device registers to the service. The generation of the Push-Token is defined as *opaque* by Apple since it is made server side, same goes for its precise usage. For now, we see it as the registration of a provider and a routing information.

As a provider because one needs to register itself as a iMessage provider, but also as a routing information because Apple needs to know where to deliver the messages. As a matter of fact, Apple fanatics users often have more than one device, and a iMessage must be delivered to all the devices at once.

So, when the client sends *one* message, it actually sends as many messages as the recipients has Push-Tokens, so that the message can be delivered to each iDevice. These messages are sent to the *Apple Push Network Service* (*APNS*) through a gateway on port 5223 as explained earlier.

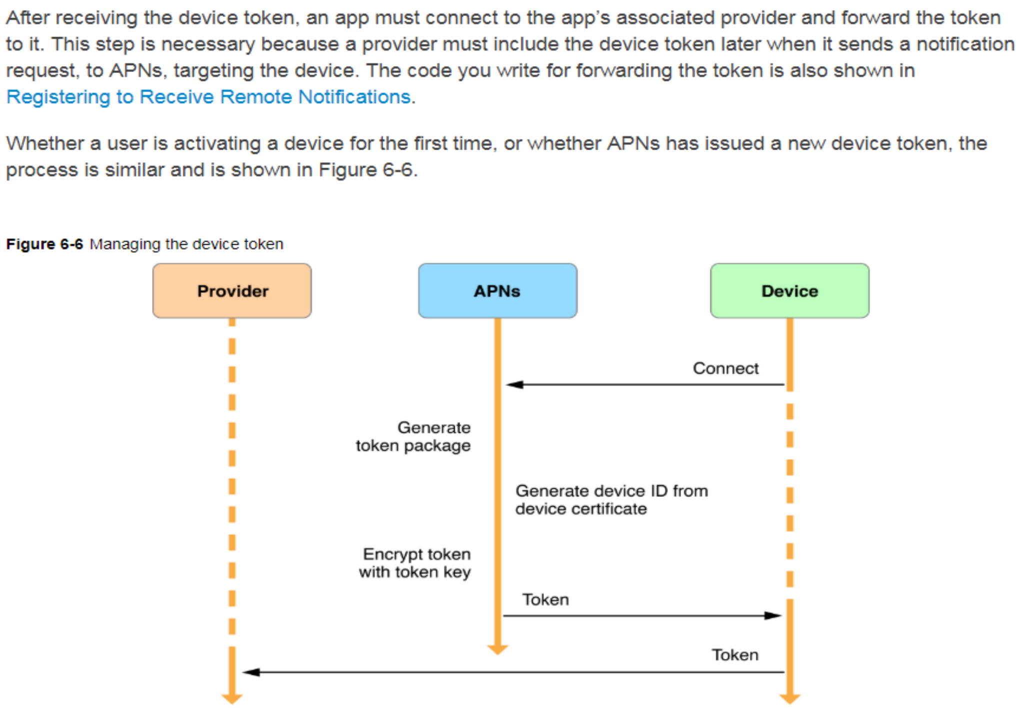
….



When the APNS servers receive that, PUSH magic comes into play. That is not described in the documentation, but we can assume, reading from [5] that Apple relies on a connection between the APNS and the devices. Based on the Push-Token for each message, Apple is able to select to what device a PUSH notification must be sent to.

https://blog.quarkslab.com/imessage-privacy.html

283. Claim 1 of the ’962 patent recites “send a second identifier to the client device in response to the received registration information.” The ’962 accused products satisfy this limitation. Specifically, the Apple server supporting APNs functionality sends a second identifier to the client device in response to the received registration information, as shown by the exemplary documentation below.



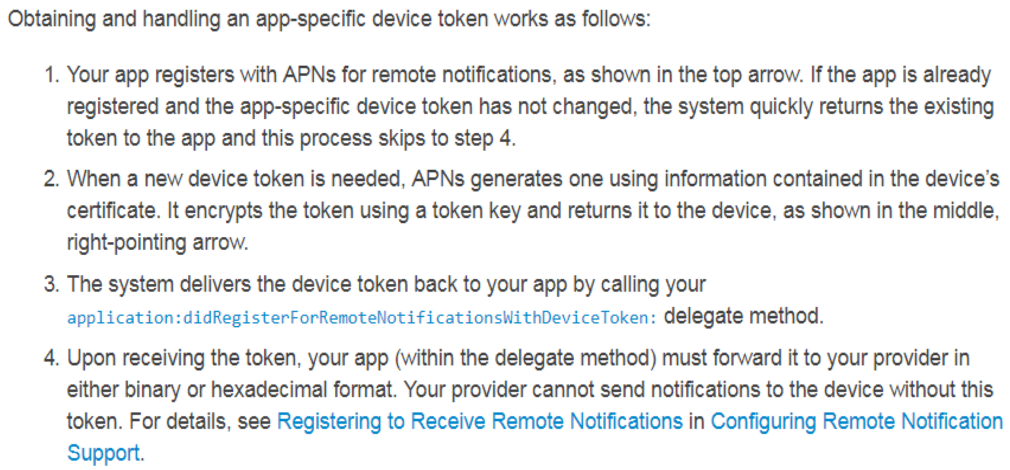
After receiving the device token, an app must connect to the app's associated provider and forward the token to it. This step is necessary because a provider must include the device token later when it sends a notification request, to APNs, targeting the device. The code you write for forwarding the token is also shown in Registering to Receive Remote Notifications.

Whether a user is activating a device for the first time, or whether APNs has issued a new device token, the process is similar and is shown in Figure 6-6.

Figure 6-6 Managing the device token

Provider
APNs
Device
Connect
Generate token package
Generate device ID from device certificate
Encrypt token with token key
Token
Token

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1



Obtaining and handling an app-specific device token works as follows:

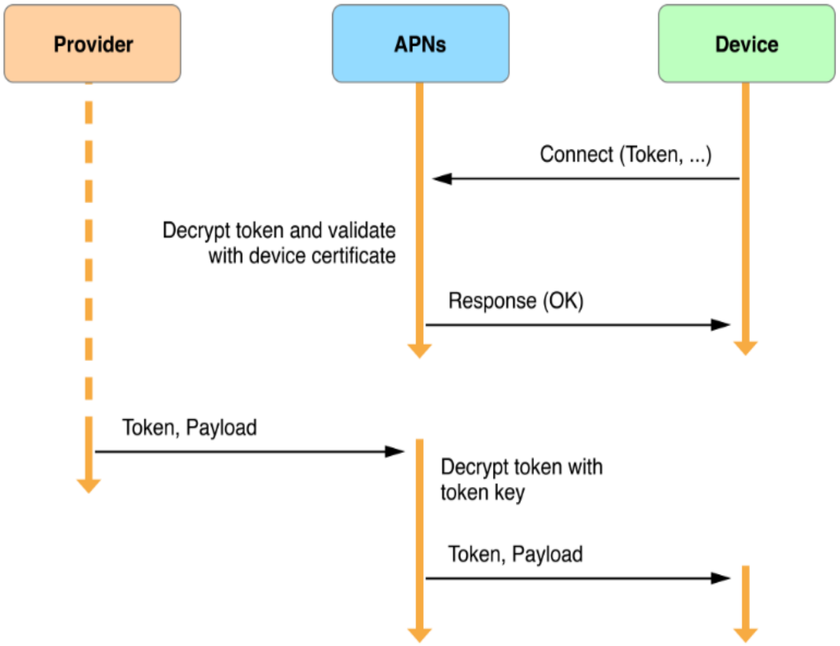
1. Your app registers with APNs for remote notifications, as shown in the top arrow. If the app is already registered and the app-specific device token has not changed, the system quickly returns the existing token to the app and this process skips to step 4.
2. When a new device token is needed, APNs generates one using information contained in the device's certificate. It encrypts the token using a token key and returns it to the device, as shown in the middle, right-pointing arrow.
3. The system delivers the device token back to your app by calling your `application:didRegisterForRemoteNotificationsWithDeviceToken:` delegate method.
4. Upon receiving the token, your app (within the delegate method) must forward it to your provider in either binary or hexadecimal format. Your provider cannot send notifications to the device without this token. For details, see Registering to Receive Remote Notifications in Configuring Remote Notification Support.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

284. Claim 1 of the '962 patent recites "configure a service to receive data from a first data store on behalf of the client device, wherein the service is based on the second identifier." The '962 accused products satisfy this limitation. The '962 accused products configure a service to receive data from the data store on behalf of the client device, wherein the service is based on

the second identifier. For example, the Apple server supporting APNs functionality configures a service to receive new information from a third party app provider on behalf of the client device, wherein the service is basen on a second identifier.

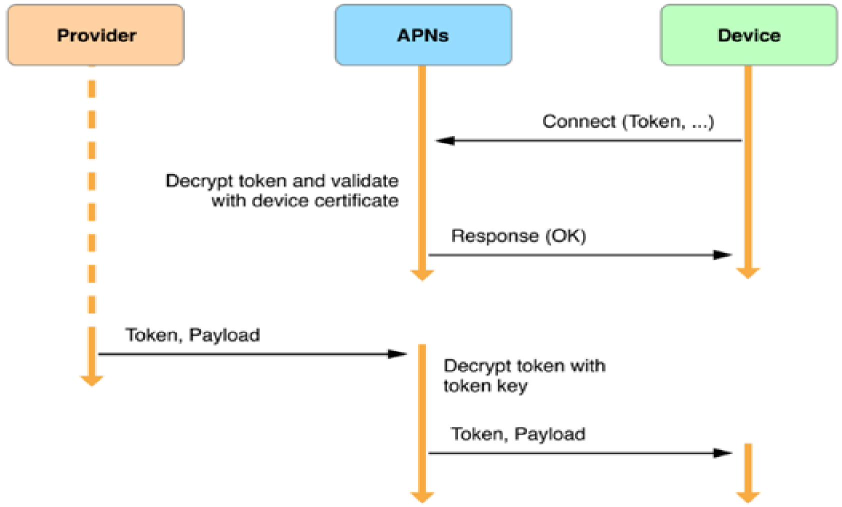
Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

285. Claim 1 of the ’962 patent recites “receive a communication from the client device to receive data from the plurality of data stores.” The ’962 accused products satisfy this limitation. The ’962 accused products receive a communication from the client device to receive data from the plurality of data stores.

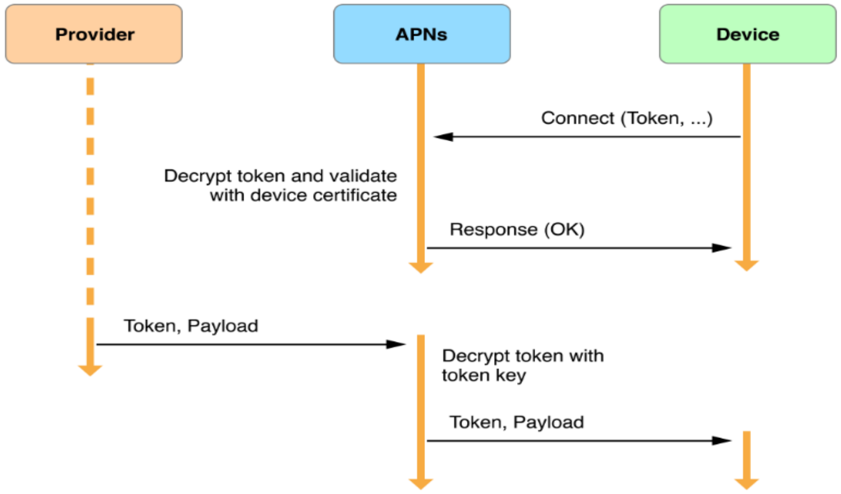
Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

286. Claim 1 of the ’962 patent recites “wherein the communication includes the first identifier.” On information and belief, the ’962 accused products satisfy this limitation. The ’962 accused products include this limitation. For example, the communication includes a first identifier, as shown by the exemplary documentation below and the iPhone logs cited above with respect to the “send a first identifier…” limitation.

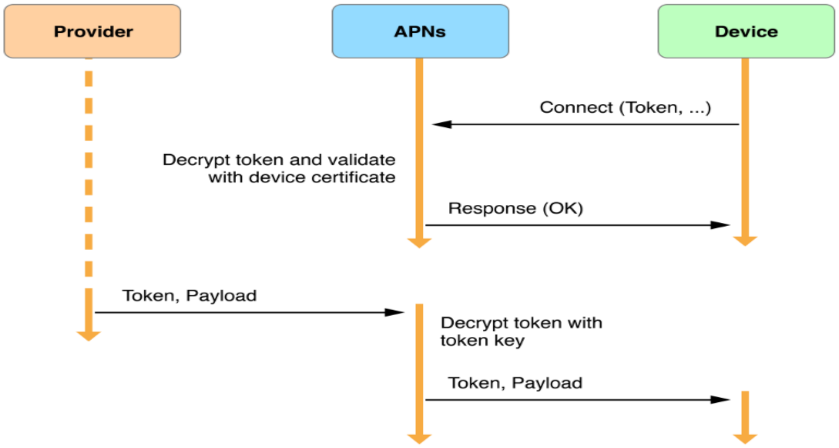
Figure 6-7 Remote notification path from provider to device



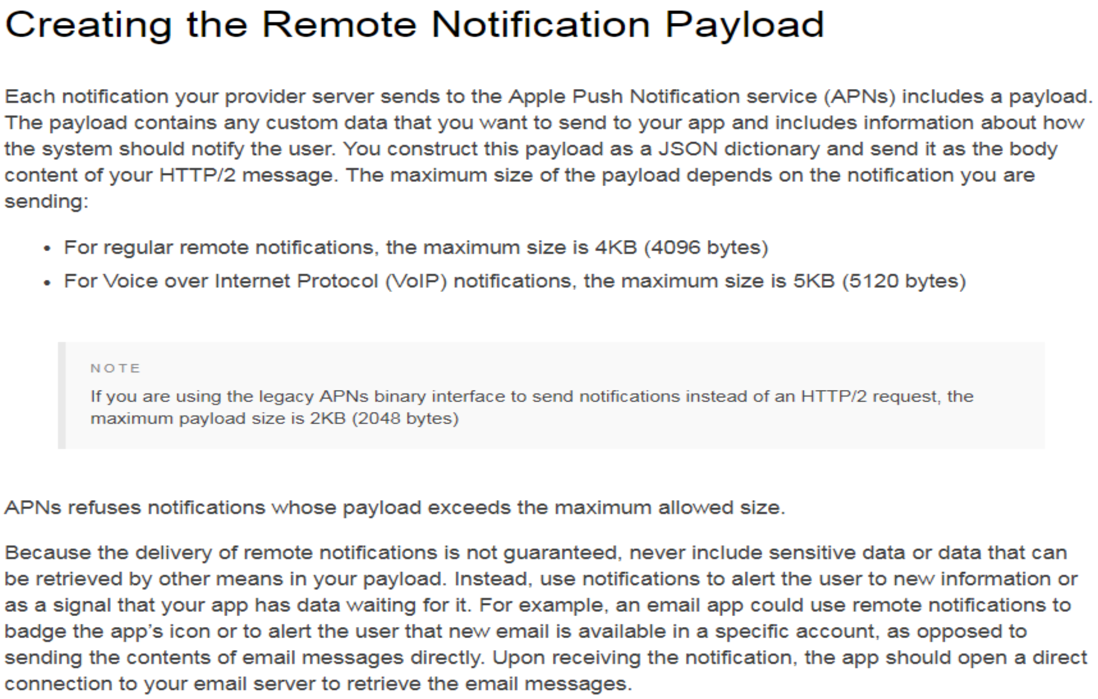
https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

287. Claim 1 of the ’962 patent recites “receive a first message from the first data store, wherein the first message is indicative of new data at the first data store.” The ’962 accused products satisfy this limitation. The ’962 accused products receive a first message from the first data store, wherein the first message is indicative of new data at the first data store, as shown by the exemplary documentation below.

Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1



Creating the Remote Notification Payload

Each notification your provider server sends to the Apple Push Notification service (APNs) includes a payload. The payload contains any custom data that you want to send to your app and includes information about how the system should notify the user. You construct this payload as a JSON dictionary and send it as the body content of your HTTP/2 message. The maximum size of the payload depends on the notification you are sending:

- For regular remote notifications, the maximum size is 4KB (4096 bytes)
- For Voice over Internet Protocol (VoIP) notifications, the maximum size is 5KB (5120 bytes)

> NOTE
>
> If you are using the legacy APNs binary interface to send notifications instead of an HTTP/2 request, the maximum payload size is 2KB (2048 bytes)

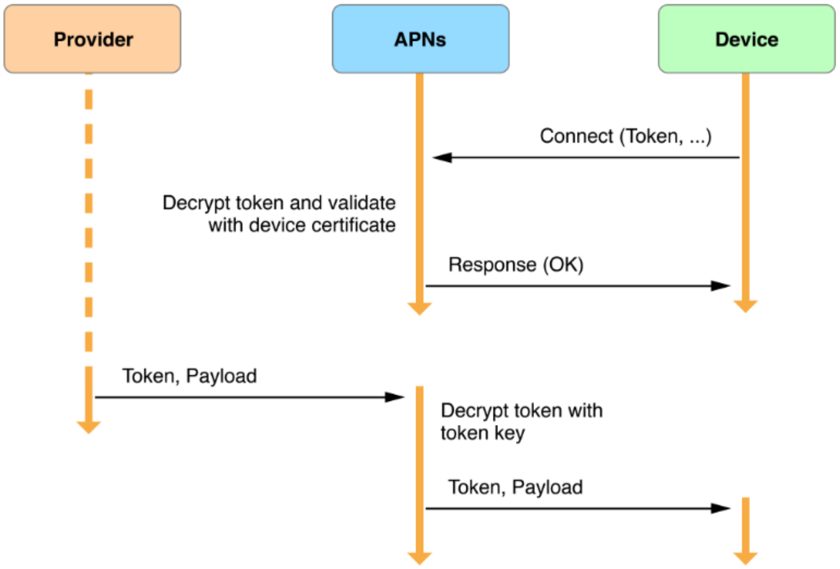
APNs refuses notifications whose payload exceeds the maximum allowed size.

Because the delivery of remote notifications is not guaranteed, never include sensitive data or data that can be retrieved by other means in your payload. Instead, use notifications to alert the user to new information or as a signal that your app has data waiting for it. For example, an email app could use remote notifications to badge the app’s icon or to alert the user that new email is available in a specific account, as opposed to sending the contents of email messages directly. Upon receiving the notification, the app should open a direct connection to your email server to retrieve the email messages.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/CreatingtheNotificationPayload.html#//apple_ref/doc/uid/TP40008194-CH10-SW1

288. Claim 1 of the ’962 patent recites “transmit a second message to the client device in response to receipt of the first message and authentication of the client device, wherein the second message is transmitted over a subsequent connection, and wherein the authentication of the client device is based on the first identifier.” The ’962 accused products satisfy this limitation. The ’962 accused products transmit a second message to the client device in response to receipt of the first message and authentication of the client device, wherein the second message is transmitted over a subsequent connection, and wherein the authentication of the client device is based on the first identifier, as shown by the exemplary documentation below as well as the iPhone logs cited above for the “send a first identifier…”

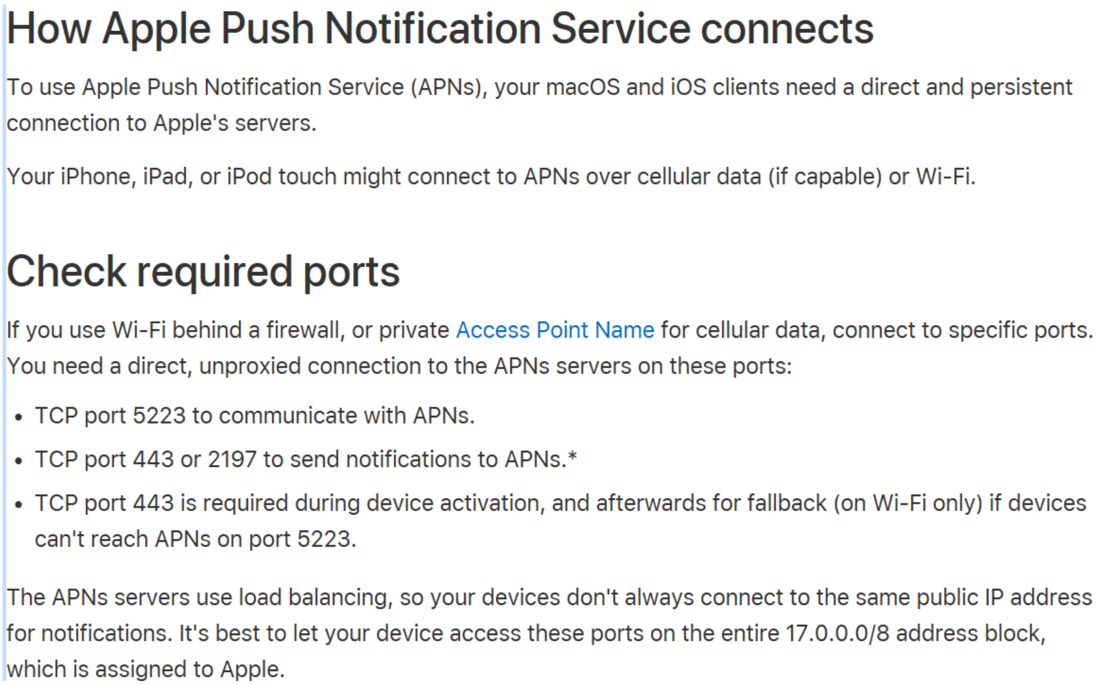
Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

289. Claim 1 of the ’962 patent recites “wherein the subsequent connection is an IP connection between the client device and the server.” The ’962 accused products satisfy this

limitation. A subsequent connection that is an IP connection is between the client device and the Apple server.



How Apple Push Notification Service connects

To use Apple Push Notification Service (APNs), your macOS and iOS clients need a direct and persistent connection to Apple's servers.

Your iPhone, iPad, or iPod touch might connect to APNs over cellular data (if capable) or Wi-Fi.

Check required ports

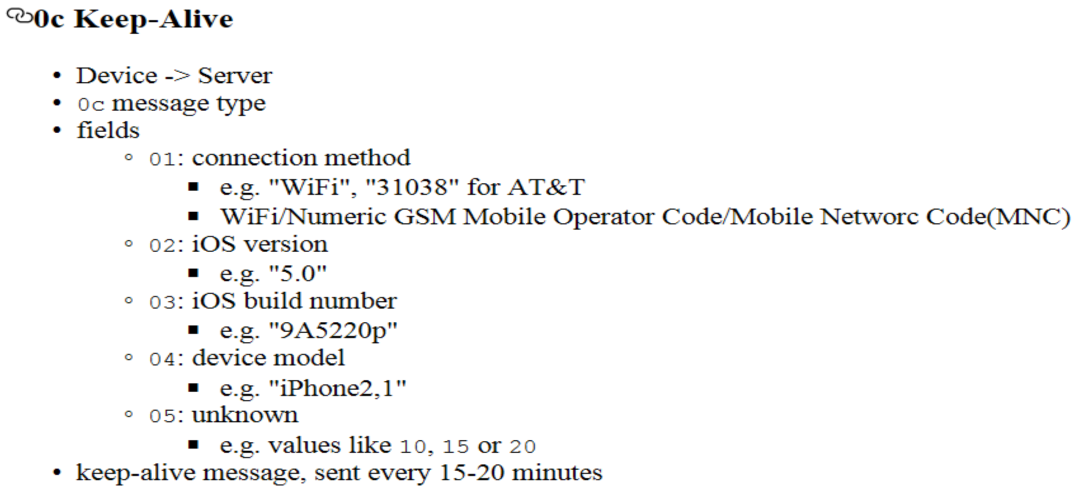
If you use Wi-Fi behind a firewall, or private Access Point Name for cellular data, connect to specific ports. You need a direct, unproxied connection to the APNs servers on these ports:

- TCP port 5223 to communicate with APNs.
- TCP port 443 or 2197 to send notifications to APNs.*
- TCP port 443 is required during device activation, and afterwards for fallback (on Wi-Fi only) if devices can't reach APNs on port 5223.

The APNs servers use load balancing, so your devices don't always connect to the same public IP address for notifications. It's best to let your device access these ports on the entire 17.0.0.0/8 address block, which is assigned to Apple.

https://support.apple.com/en-us/HT203609

290. Claim 1 of the ’962 patent recites “receive a keep-alive message from the client device for maintenance of the subsequent connection.” The ’962 accused products satisfy this limitation. The ’962 accused products receive a keep-alive message from the client device for maintenance of the subsequent connection, as shown by the exemplary documentation below.

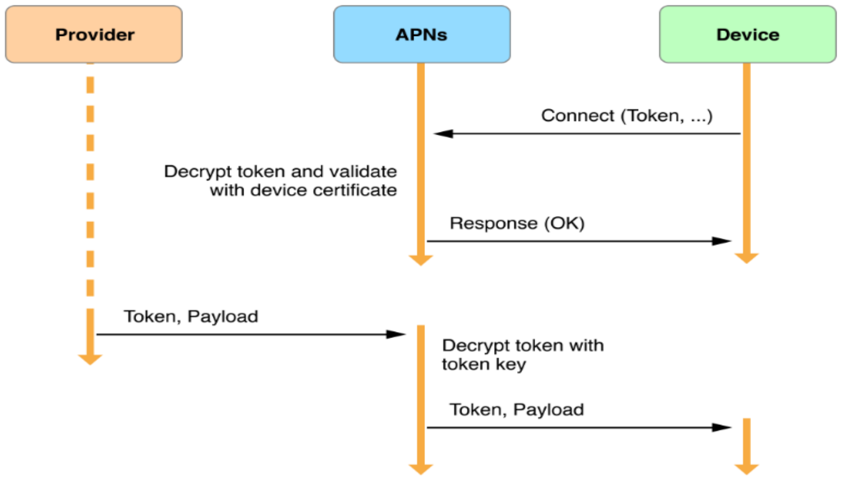


0c Keep-Alive

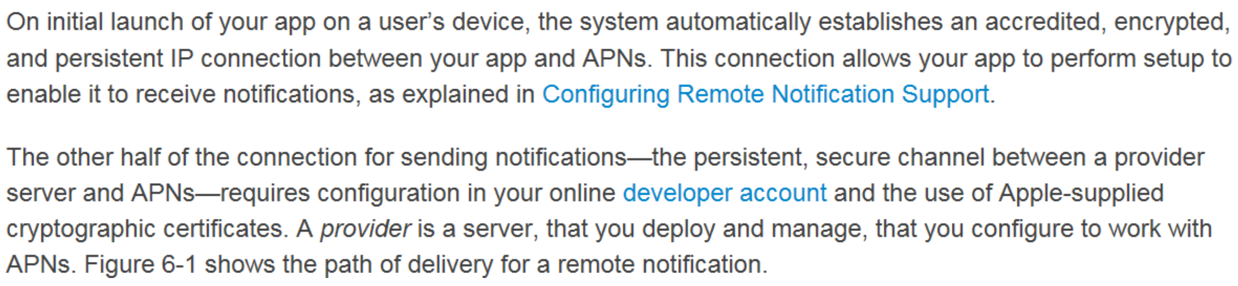
- Device -> Server
- `0c` message type
- fields
  - `01`: connection method
    - e.g. "WiFi", "31038" for AT&T
    - WiFi/Numeric GSM Mobile Operator Code/Mobile Networc Code(MNC)
  - `02`: iOS version
    - e.g. "5.0"
  - `03`: iOS build number
    - e.g. "9A5220p"
  - `04`: device model
    - e.g. "iPhone2,1"
  - `05`: unknown
    - e.g. values like `10`, `15` or `20`
- keep-alive message, sent every 15-20 minutes

https://github.com/meeee/pushproxy/blob/master/doc/apple-push-protocol-ios5-lion.md

Figure 6-7 Remote notification path from provider to device



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

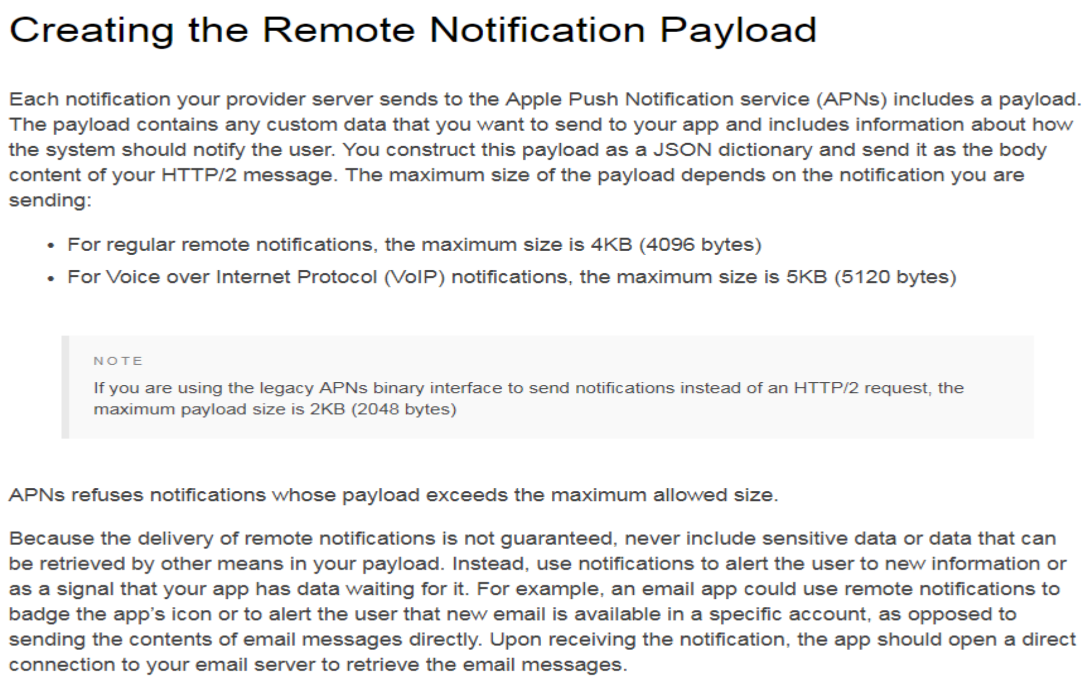


On initial launch of your app on a user's device, the system automatically establishes an accredited, encrypted, and persistent IP connection between your app and APNs. This connection allows your app to perform setup to enable it to receive notifications, as explained in Configuring Remote Notification Support.

The other half of the connection for sending notifications—the persistent, secure channel between a provider server and APNs—requires configuration in your online developer account and the use of Apple-supplied cryptographic certificates. A *provider* is a server, that you deploy and manage, that you configure to work with APNs. Figure 6-1 shows the path of delivery for a remote notification.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

291. Claim 1 of the '962 patent recites "wherein additional information associated with the first message is sent from the first data store to the client device upon receipt of the second message by the client device." The '962 accused products satisfy this limitation. Additional information associated with the first message is sent from the first data store to the client device upon receipt of the second message by the client device, as shown by the exemplary documentation below.



Creating the Remote Notification Payload

Each notification your provider server sends to the Apple Push Notification service (APNs) includes a payload. The payload contains any custom data that you want to send to your app and includes information about how the system should notify the user. You construct this payload as a JSON dictionary and send it as the body content of your HTTP/2 message. The maximum size of the payload depends on the notification you are sending:

- For regular remote notifications, the maximum size is 4KB (4096 bytes)
- For Voice over Internet Protocol (VoIP) notifications, the maximum size is 5KB (5120 bytes)

> NOTE
>
> If you are using the legacy APNs binary interface to send notifications instead of an HTTP/2 request, the maximum payload size is 2KB (2048 bytes)

APNs refuses notifications whose payload exceeds the maximum allowed size.

Because the delivery of remote notifications is not guaranteed, never include sensitive data or data that can be retrieved by other means in your payload. Instead, use notifications to alert the user to new information or as a signal that your app has data waiting for it. For example, an email app could use remote notifications to badge the app's icon or to alert the user that new email is available in a specific account, as opposed to sending the contents of email messages directly. Upon receiving the notification, the app should open a direct connection to your email server to retrieve the email messages.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/CreatingtheNotificationPayload.html#//apple_ref/doc/uid/TP40008194-CH10-SW1

292. Thus, as illustrated above, the '962 accused products directly infringe one or more claims of the '962 patent. Apple makes, uses, sells, offers for sale, exports, and/or imports, in this district and/or elsewhere in the United States, these devices and thus directly infringes the '962 patent.

293. Apple indirectly infringes the '962 patent, as provided in 35 U.S.C. § 271(b) and/or 271(c), including by inducing infringement by others, such as Apple's customers and end-users, in this district and elsewhere in the United States. For example, Apple's customers and end-users directly infringe through their use of the inventions claimed in the '962 patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the '962 accused products, and providing instructions, documentation, and other information to customers and end-users suggesting they use the '962 accused products in an infringing manner, including in-store technical support,

online technical support, marketing, product manuals, advertisements, online documentation, marketing materials, technical specifications, data sheets, web pages on its website (e.g., www.apple.com), press releases, user manuals, and trade shows (e.g., CES and Mobile World Congress), including all of the Apple documentation cited above as exemplary evidence of infringement. By way of example, Apple user manuals and documentation cited above instruct, promote, and encourage the use of the accused products' capability in an infringing manner, including through the use of Apple's APN service. *See, e.g.*, https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1; https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/CreatingtheNotificationPayload.html#//apple_ref/doc/uid/TP40008194-CH10-SW1. As a result of Apple's inducement, Apple's customers and end-users use the Apple products in the way Apple intends and directly infringe the '962 patent. Apple performs these affirmative acts with knowledge of the '962 patent and with the intent, or willful blindness, that the induced acts directly infringe the '962 patent. Apple has had knowledge and notice of the '962 patent at least as of the filing of this complaint.

294. Apple, by way of its infringing activities, has caused and continues to cause Plaintiff to suffer damages, the exact amount to be determined at trial.

VI. PRAYER FOR RELIEF

WHEREFORE, Plaintiff prays for the following relief:

295. A judgment in favor of Plaintiff that Apple, has infringed, directly and indirectly, including by way of inducement infringement, literally and/or under the doctrine of equivalents, the patents-in-suit;

296. Plaintiff's actual damages in an amount sufficient to compensate Plaintiff for Apple's infringement of the patents-in-suit until such time as Apple ceases its infringing conduct, including supplemental damages post-verdict;

297. Pre- and post-judgment interest as allowed by law on any damages awarded to Plaintiff;

298. A judgment and order requiring Apple to pay the expenses and costs of this action (including all disbursements), as well as attorneys' fees as provided by 35 U.S.C. § 285;

299. A judgment and order requiring that Apple pay to Plaintiff compulsory ongoing licensing fees, as determined by the Court in equity; and

300. Such other and further relief in law or in equity to which Plaintiff may be justly entitled.

VII. DEMAND FOR JURY TRIAL

Plaintiff demands a trial by jury of any and all issues triable of right before a jury, except for future patent infringement, which is an issue in equity to be determined by the Court.

Dated: April 10, 2019.

MCKOOL SMITH, P.C.

/s/ Sam Baxter
Samuel F. Baxter
Lead Attorney
Texas State Bar No. 01938000
sbaxter@mckoolsmith.com
Jennifer Truelove
Texas State Bar No. 24012906
jtruelove@mckoolsmith.com
MCKOOL SMITH, P.C.
104 East Houston, Suite 300
Marshall, Texas 75670
Telephone: (903) 923-9000
Facsimile: (903) 923-9099

Kevin Burgess
Texas State Bar No. 24006927
kburgess@mckoolsmith.com
Seth R. Hasenour
Texas State Bar No. 24059910
shasenour@mckoolsmith.com
MCKOOL SMITH, P.C.
300 W. 6th Street, Suite 1700
Austin, Texas 78701
Telephone: (512) 692-8704

Kevin Schubert
kschubert@mckoolsmith.com
MCKOOL SMITH, P.C.
One Bryant Park, 47th Floor
New York, NY 10036
Telephone: (212) 402-9400

Eric Hansen
Texas State Bar No. 19196650
ehansen@mckoolsmith.com
MCKOOL SMITH, P.C.
300 Crescent Court, Suite 1500
Dallas, Texas 75201
Telephone: (214) 978-4000

ATTORNEYS FOR PLAINTIFF SEVEN NETWORKS, LLC